

University of Warwick institutional repository: <http://go.warwick.ac.uk/wrap>

A Thesis Submitted for the Degree of EngD at the University of Warwick

<http://go.warwick.ac.uk/wrap/3023>

This thesis is made available online and is protected by original copyright.

Please scroll down to view the document itself.

Please refer to the repository record for this item for information to help you to cite it. Our policy information is available from the repository home page.

Protocol Security for Third Generation Telecommunication Systems

by

Theodore Stergiou

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor
of Philosophy in Engineering

University of Warwick, School of Engineering

August 2003

*To those I've met, and those I will,
For they are writing my very own book*

Table of Contents

List of Figures and Tables	iii
Acknowledgments.....	vi
List of Abbreviations	vii
Thesis Summary	ix
Chapter 1. Thesis Review	2
1.1 Overview	3
1.2 Methodology	4
1.3 Original Contributions	7
1.4 Terminology	8
1.5 Thesis Organisation	9
Chapter 2. Communication Systems	12
2.1 Overview	13
2.1.1 Circuit-Switched Systems	13
2.1.2 Packet-Switched Systems.....	15
2.2 Wireless Telecommunication Network Systems	19
2.2.1 Overview	19
2.2.2 Supporting Protocols.....	22
Chapter 3. Network Protocol Security.....	32
3.1 Overview	33
3.2 Network Attacks.....	34
3.3 Protocol Security.....	38
3.3.1 Architectural Analysis.....	39
3.3.2 OSI Security Model	50
Chapter 4. Future Core Networks System (FCNS) – Communication Layers.....	58
4.1 Overview	59
4.2 FCNS Architectural View.....	60
4.3 FCNS Functional View	65
4.3.1 User-Defined Presentation (UDPRES) Layer.....	66
4.3.2 User-Defined Session (UDSES) Layer	76
4.3.3 Transmission Layer (TX_LAYER)	88
4.3.4 End-to-End Layer (EE_LAYER).....	106
4.3.5 Physical Layer (PHYS)	123
Chapter 5. Future Core Networks System (FCNS) – Security Layer Protocol	139
5.1 Overview	140
5.2 FCNS Security Layer (SL)	141
5.2.1 SL Protocol Definition.....	142
5.2.2 SL Service Definition.....	153
5.3 FCNS Keystream Generator	160
Chapter 6. Future Core Networks System Error Protocol (FCNSEP)	172
6.1 Overview	173
6.2 Architectural Analysis	175
6.3 FCNSEP Functionality.....	178
6.4 Security Considerations	186
6.5 Applicability.....	192
Chapter 7. Future Core Networks System Implementation	195
7.1 Overview	196
7.2 FCNS Design Process	197
7.3 FCNS Prototype.....	208
7.4 Source Code Implementation.....	212
Chapter 8. Case Studies	220
8.1 Overview	221
8.2 FCNS Stack Implementation	221
8.2.1 Simulation Environment Description.....	222
8.2.2 Results and Measurements	225
8.3 Packet – Switched Architecture	249
8.3.1 Simulation Environment Description.....	250
8.3.2 Results and Measurements	251

Chapter 9. Conclusions and Suggestions for Future Work	273
9.1 Overview	274
9.2 FCNS Applicability	279
9.3 Further Work	283
Appendix A. FCNS Protocol Specification Diagrams	287
Figure A.1 – UMTS CN R99.....	287
Figure A.2.1 – Node Sending Instance.....	288
Figure A.2.2 – Node Receiving Instance.....	289
Figure A.3.1 – UDPRES Sending Instance	290
Figure A.3.2 – UDPRES Receiving Instance	291
Figure A.4.1 – UDSES Sending Instance	292
Figure A.4.2 – UDSES Receiving Instance	293
Figure A.5.1 – TX_LAYER Sending Instance.....	294
Figure A.5.2 – TX_LAYER Receiving Instance.....	295
Figure A.6.1 – EE_LAYER Sending Instance.....	296
Figure A.6.2 – EE_LAYER Receiving Instance.....	297
Figure A.7.1 – PHYS Sending Instance.....	298
Figure A.7.2 – PHYS Receiving Instance.....	299
Figure A.8.1 – SL Sending Instance	300
Figure A.8.2 – SL Receiving Instance	301
Figure A.9.1 – FCNSEP Sending Instance	302
Figure A.9.2 – FCNSEP Receiving Instance.....	303
Appendix B. FCNS Keystream Generator Statistical Test Results	304
B.1.1 MWCG of Equation 5-1 – 1 st Test Set	306
B.1.2 MWCG of Equation 5-1 – 2 nd Test Set.....	311
B.1.3 MWCG of Equation 5-2 – 1 st Test Set	317
B.1.4 MWCG of Equation 5-2 – 2 nd Test Set.....	322
Appendix C. List of Publications	328
References & Bibliography	330

List of Figures and Tables

Figure 1.1: Protocol design and development steps.....	5
Figure 2.1: SS7 architecture and its relation to OSI	14
Figure 2.2: Circuit Switching vs. Packet Switching	16
Table 2.1: Characteristics of the CS and PS	17
Figure 2.3: Digital communications system.....	18
Figure 2.4: Basic UMTS connectivity concept.....	20
Table 2.2: UMTS integration domains.....	20
Figure 2.5: ATM reference model.....	23
Figure 2.6: SCTP message format.....	25
Figure 2.7: IPv6 packet header	26
Table 2.3: IPv6 extension headers.....	27
Figure 2.8: TCP message structure	28
Table 2.4: TCP flag field meanings.....	29
Figure 3.1: Classification of telecommunication network attacks.....	35
Figure 3.2: ATM Security Information Element	40
Figure 3.3: AH header format	45
Figure 3.4: ESP header format	45
Figure 3.5: Layer N security architecture.....	51
Figure 3.6: OSI security model mechanisms	52
Figure 3.7: OSI security functionality	52
Figure 3.8: FCNS Layer N Security Architecture.....	55
Figure 4.1: FCNS conceptual view.....	60
Figure 4.2: FCNS and OSI model	60
Figure 4.3: Two peers communicating using the FCNS	62
Figure 4.4: FCNS internal communication structure	63
Table 4.1: Decisions and decision responses of the UDPRES protocol	67
Table 4.2: Service primitives exchanged by the UDPRES protocol	67
Figure 4.5: NODE_REQUEST message.....	71
Figure 4.6: NODE_REQUEST_RESPONSE message.....	71
Figure 4.7: NODE_ESTABLISH message	71
Figure 4.8: UDPRES NODE_TRANSFER message.....	74
Figure 4.9: NODE_ALERT message	74
Figure 4.10: NODE_ABORT message.....	74
Figure 4.11: NODE_END message	76
Table 4.3: FCNS QoS parameters	82
Figure 4.12: UDSES MSG_CREATE message	84
Figure 4.13: SET_STATUS message.....	86
Figure 4.14: ENCRYPT_ETE and DECRYPT_ETE messages	96
Table 4.4: TX_LAYER QoS parameters.....	98
Figure 4.15: DATAGRAM and DATAGRAM_SAR structure	100
Table 4.5: EE_LAYER recoverable / signalled errors	112
Table 4.6: Reasons for rejecting address-formatting requests	116
Table 4.7: EE_LAYER datagram rejection parameters values	118
Figure 4.16: PACKET message structure.....	119
Figure 4.17: MSG_FINAL message structure	120
Figure 4.18: PHYS protocol frame structure	133
Figure 5.1: Sample communication attacks.....	144
Figure 5.2: Sample attack against message confidentiality	146
Figure 5.3: FCNS 10-way handshake mechanism.....	150
Table 5.1: Message explanation for the FCNS 10-way handshake function	151
Figure 5.4: FCNS Keystream generator architectural view.....	161
Table 5.2: Required length intervals for the Runs statistical test.....	165
Figure 6.1: FCNSEP realisation in the FCNS stack architecture.....	174
Figure 6.2: ERROR_REP FCNSEP message structure.....	175
Figure 6.3: ERROR_RESP FCNSEP message structure.....	176
Table 6.1: Error conditions for the FCNSEP ERROR_REPORT message field	176
Table 6.2: Sample suggested error correction and recovery procedures.....	180

Table 6.3: ERROR_REP erroneous reception conditions.....	183
Figure 6.4: NAK_REP message structure	183
Figure 6.5: ICMPv6 header message structure	189
Table 6.4: Destination unreachable ICMP notification instances	191
Figure 6.6: Sample integrated communications network.....	193
Table 7.1: FCNS external messages.....	202
Table 7.2: FCNS internal messages.....	203
Figure 7.1: Relative FCNS protocol efficiency	207
Figure 7.2: FCNS prototype model architecture	208
Table 7.3: Memory requirements of the XSpin verification tool for the FCNS model	211
Figure 7.3: FCNS stack simulation environment.....	213
Table 7.4: Message sending processing time.....	216
Table 7.5: FCNS simulation environment memory consumption.....	218
Figure 8.1: FCNS OMNET++ simulation topology.....	222
Table 8.1: FCNS communication layer header sizes.....	223
Table 8.2: TCP/IPv4 protocol suite header sizes	223
Table 8.3: TCP/IPv6 protocol suite header sizes	224
Figure 8.2: Congestion effects on packet throughput	226
Figure 8.3: Throughput variations – FCNS full security measures.....	228
Figure 8.4: Reference loss ratio for RTT of 2.5 msec and variable BER	229
Figure 8.5: Reference FCNS packet throughput for RTT of 2.5 msec and variable BER	230
Figure 8.6: Throughput comparison – reference RTT 2.5 msec and theoretical calculation	231
Figure 8.7: Loss ratio – RTT 2.5 msec and variable BER	233
Figure 8.8: EE_LAYER packet throughput – RTT 2.5 msec and variable BER.....	235
Figure 8.9: Frame throughput – RTT 2.5 msec and variable BER	236
Figure 8.10: Loss ratio – RTT of 60 msec and variable BER.....	237
Figure 8.11: Loss Ratio – RTT 60 msec, comparison with FCNSEP realisation	238
Figure 8.12: EE_LAYER packet throughput response comparisons.....	240
Figure 8.13: Frame throughput – RTT 60 msec and variable BER.....	241
Table 8.4: Delay variations (jitter) with respect to OMNET++ timing.....	241
Figure 8.14: Loss ratio – Variable RTT and BER, FCNS realisation comparison	242
Figure 8.15: Frame throughput – Variable RTT and BER, FCNS realisation comparison.....	243
Figure 8.16: Relative FCNS efficiency, FCNS realisation comparison for variable data size	244
Figure 8.17: Relative FCNS Efficiency, FCNS realisation comparison for variable BER	245
Figure 8.18: Loss Ratio – Variable delay, BER greater than 10^{-6}	246
Figure 8.19: EE_LAYER packet throughput – Variable delay, BER greater than 10^{-6}	247
Figure 8.20: Frame throughput – Variable delay, BER less than 10^{-4}	248
Figure 8.21: Packet-switched network OMNET++ simulation topology.....	250
Figure 8.22: Reference loss ratio – RTT of 100 msec and variable BER.....	252
Figure 8.23: Reference EE_LAYER packet throughput – RTT of 100 msec and variable BER	253
Figure 8.24: Reference frame throughput – RTT of 100 msec and variable BER..	254
Figure 8.25: Loss Ratio – RTT of 200 msec and variable BER	256
Figure 8.26: EE_LAYER packet throughput – RTT of 200 msec and variable BER.	257
Figure 8.27: Frame throughput – RTT of 200 msec and variable BER	258
Figure 8.28: Loss ratio – RTT of 260 msec and variable BER	259
Figure 8.29: EE_LAYER packet throughput – RTT of 260 msec and variable BER.	260
Figure 8.30: Frame throughput – RTT of 260 msec and variable BER	261
Figure 8.31: FCNS vs. TCP/IPv4 – Constant data size 1460 bytes and variable BER	263
Figure 8.32: FCNS vs. TCP/IPv6 – Constant data size 1460 bytes and variable BER	264

Figure 8.33: FCNS vs. TCP/IP – Constant BER (8×10^{-6}) and variable data size....	265
Figure 8.34: FCNS vs. TCP/IP – Constant BER (10^{-5}) and variable data size	266
Figure 8.35: FCNS vs. TCP/IPv4 – Variable data size throughput responses	267
Figure 8.36: FCNS vs. TCP/IPv6 – Variable data size throughput responses	268
Figure 8.37: FCNS vs. TCP/IPv4 – Variable data size throughput responses acknowledging frame blocks	269
Figure 8.38: FCNS vs. TCP/IPv6 – Variable data size throughput responses acknowledging frame blocks	270
Figure 8.39: FCNS vs. TCP/IPv6 – throughput efficiency comparison.....	271
Figure B.1.1: Standard normal pdf.....	305
Figure B.1.2: Standard uniform cdf.....	305
Figure B.2.1: Frequency test normal pdf – first MWCG, first test.....	306
Figure B.2.2: Frequency test uniform cdf – first MWCG, first test.....	307
Figure B.3.1: Poker test normal pdf – first MWCG, first test	307
Figure B.3.2: Poker test uniform cdf – first MWCG, first test	308
Figure B.4.1: Runs and long runs tests normal pdf – first MWCG, first test	308
Figure B.4.2: Runs and long runs tests uniform cdf – first MWCG, first test.....	309
Figure B.5.1: Birthday test normal pdf – first MWCG, first test	309
Figure B.5.2: Birthday test uniform cdf – first MWCG, first test	310
Figure B.6.1: Bitstream test normal pdf – first MWCG, first test	310
Figure B.6.2: Bitstream test uniform cdf – first MWCG, first test.....	311
Figure B.7.1: Frequency test normal pdf – first MWCG, second test	312
Figure B.7.2: Frequency test uniform cdf – first MWCG, second test.....	312
Figure B.8.1: Poker test normal pdf – first MWCG, second test.....	313
Figure B.8.2: Poker test uniform cdf – first MWCG, second test	313
Figure B.9.1: Runs and long runs tests normal pdf – first MWCG, second test	314
Figure B.9.2: Runs and long runs test uniform cdf – first MWCG, second test.....	314
Figure B.10.1: Birthday test normal pdf – first MWCG, second test	315
Figure B.10.2: Birthday test uniform cdf – first MWCG, second test	315
Figure B.11.1: Bitstream test normal pdf – first MWCG, second test	316
Figure B.11.2: Bitstream test uniform cdf – first MWCG, second test.....	316
Figure B.12.1: Frequency test normal pdf – second MWCG, first test.....	317
Figure B.12.2: Frequency test uniform cdf – second MWCG, first test.....	318
Figure B.13.1: Poker test normal pdf – second MWCG, first test	318
Figure B.13.2: Poker test uniform cdf – second MWCG, first test	319
Figure B.14.1: Runs and long runs tests normal pdf – second MWCG, first test ..	319
Figure B.14.2: Runs and long runs test uniform cdf – second MWCG, first test ...	320
Figure B.15.1: Birthday test normal pdf – second MWCG, first test	320
Figure B.15.2: Birthday test uniform cdf – second MWCG, first test	321
Figure B.16.1: Bitstream test normal pdf – second MWCG, first test	321
Figure B.16.2: Bitstream test uniform cdf – second MWCG, first test.....	322
Figure B.17.1: Frequency test normal pdf – second MWCG, second test	323
Figure B.17.2: Frequency test uniform cdf – second MWCG, second test	323
Figure B.18.1: Poker test normal pdf – second MWCG, second test.....	324
Figure B.18.2: Poker test uniform cdf – second MWCG, second test.....	324
Figure B.19.1: Runs and long runs test normal pdf – second MWCG, second test	325
Figure B.19.2: Runs and long runs test uniform cdf – second MWCG, second test	325
Figure B.20.1: Birthday test normal pdf – second MWCG, second test.....	326
Figure B.20.2: Birthday test uniform cdf – second MWCG, second test	326
Figure B.21.1: Bitstream test normal pdf – second MWCG, second test	327
Figure B.21.2: Bitstream test uniform cdf – second MWCG, second test	327

Acknowledgments

I would like to thank my supervisors Professor Roger Green and Dr. Mark Leeson for their help and guidance during this work. Their support and advice have been invaluable for the continuation and submission of this thesis.

I would also like to express my gratitude to my friends and family for their moral support. Their help has been vital to the pursuing of my PhD degree.

Finally, I would like to thank all those people whose conversation with has enabled me to enhance my knowledge and expand my research field of study. Their questions, comments and suggestions enabled me to refine the perspectives of this work.

List of Abbreviations

1G	First Generation
2G	Second Generation
3G	Third Generation
3GPP	Third Generation Partnership Program
AAL	ATM Adaptation Layer
AH	Authentication Header
AMPS	Advanced Mobile Phone Service
ARQ	Automatic Repeat Request
ASN.1	Abstract Syntax Notation 1
ATM	Asynchronous Transfer Mode
ATMsec	ATM security architecture
AuC	Authentication Centre
BER	Bit Error Rate
B-ISDN	Broadband ISDN
BW	Bandwidth
CA	Certification Authority
cdf	cumulative density function
CN	Core Network
CRC	Cyclic Redundancy Check code
CS	Circuit Switched
CU	Channel Utilisation
DDoS	Distributed DoS
DoS	Denial of Service
EE_LAYER	End-to-End Layer (FCNS architecture)
EIR	Equipment Identity Register
EOF	End-Of-File message identifier
ESP	Encapsulating Security Payload header
ETSI	European Telecommunications Standards Institute
FCNS	Future Core Networks System
FCNSEP	FCNS Error Protocol
FIPS	Federal Information Processing Standard
GGSN	Gateway GPRS Support Node
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
HLR	Home Location Register
ICMP	Internet Control Message Protocol
IE	Information Element
IKE	Internet Key Exchange
IP	Internet Protocol
IPsec	IP security architecture
ISAKMP	Internet Security Association and Key Management Protocol
ISDN	Integrated Services Digital Network
ISO	International Standardisation Body
IT	Information Technology
ITU	International Telecommunications Union
IV	Initialisation Vector
LAN	Local Area Network
MAC	Medium Access Control
MAP	Mobile Application Part
MAPSEC	MAP Security architecture
MFS	Maximum number of Frames per Second
MTU	Maximum Transfer Unit
MWCG	Multiplicative With Carry Generator
NDS	Network Domain Security
NIST	National Institute of Science and Technology
NMT	Nordic Mobile Telephone

NNI	Network-to-Network Interface
OAM	Operations And Maintenance (ATM cell)
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
PDP	Packet Data Protocol (e.g. FCNS, IP)
PDU	Protocol Data Unit
PHYS	Physical layer (FCNS architecture)
PRNG	Pseudo-Random Number Generator
PS	Packet Switched
PSTN	Public Switched Telephone Network
QoS	Quality of Service
R99	Release 1999 (UMTS CN architectural release)
RF	Radio Frequency
RTT	Round Trip Time
RWC	Recursion With Carry generator
SA	Security Association
SAD	Security Association Database
SC	Security Context
SCTP	Stream Control Transmission Protocol
SDH	Synchronous Digital Hierarchy
SDL	Specification and Description Language
SDP	Security Policy Database
SDU	Service Data Unit
SGSN	Serving GPRS Support Node
SHA	Secure Hash Algorithm
SL	Security Layer (FCNS architecture)
SS7	Signalling System 7
SSL	Secure Socket Layer
S-UMTS	Satellite Universal Mobile Telecommunications System component
T_{1F}	Time to transmit a single Frame
TACS	Total Access Communication System
TCP	Transmission (or Transport) Control Protocol
TX_LAYER	Transmission Layer (FCNS architecture)
UDPRES	User-Defined Presentation Layer (FCNS architecture)
UDSES	User-Defined Session Layer (FCNS architecture)
UML	Unified Modelling Language
UMTS	Universal Mobile Telecommunications System
UNI	User-to-Network Interface
UTRAN	UMTS Terrestrial Access Network
VC	Virtual Channel
VPN	Virtual Private Network
WAN	Wide Area Network
WAP	Wireless Application Protocol
W-CDMA	Wideband Code Division Multiple Access

Thesis Summary

In this thesis, a novel protocol stack architecture is presented. The Future Core Networks System (FCNS) forms a secure reference model for use in packet-switched structures, with its applicability ranging from computer to telecommunication networks. An insight on currently used network protocol systems is given, analysing standardised sets of communication rules with respect to the security they afford to the messages exchanged. The lack of protection schemes for the internal protocol stack messages and the implementation pitfalls of their security architectures are described, in relation to the effects they have on the communication process. The OSI security model is also considered, with disadvantages identified in the placement of security functionality and its management. The drawbacks depicted for currently used systems form the motivation behind this work. The analysis of the FCNS follows, which is composed of three parts. In the first part, the FCNS communication layers are examined, with respect to the mechanisms used to establish, maintain and tear down a connection between peer entities. In the second part, the security mechanisms of the proposed reference architecture are given, including details on the FCNS keystream generator used for the security of the internal FCNS messages. Finally, the FCNS Error Protocol is depicted, illustrating the modes of operation and advantages it exhibits over currently used systems. The work then moves into presenting details of the software FCNS implementation, followed by the presentation of the results and measurements obtained by the case studies created. Comparisons are given in relation to the TCP/IP suite, to provide the means of identifying the FCNS applicability in various network environments. The work is concluded by presenting the FCNS functionality in delivering information for the UMTS, together with further work that may enhance the flexibility and use of the proposed architecture.

Chapter 1: Thesis Review

Chapter 1. Thesis Review

1.1 Overview

1.2 Methodology

1.3 Original Contributions

1.4 Terminology

1.5 Thesis Organisation

A general review of the thesis and its structure with respect to the chapters presented and the applicability of the subject developed is provided. An overview of the system designed is given, together with the methodology used to materialise its software implementation. The objectives and contributions of the thesis are consequently presented, together with the various terms used throughout the document and, finally, a brief description of the chapters' contents.

1.1 Overview

Telecommunication systems management is a concept intended to decompose the management of a system into functions, enabling operators to better handle and identify potential network problems [1]. These functions include the performance management of a system, fault management, configuration management, accounting management and security management. The last of these has become a critical issue for the scientific community, due to the increased amount of vital information exchanged over the existing platforms between users and the number of subscribers requesting connections. The introduction of the Third Generation (3G) systems [2] will provide customers with advanced services and mobile commerce capabilities, further complicating the degree of effort and technology required to efficiently handle security management issues.

This thesis considers the security organization of packet-switched (PS) networks and the Universal Mobile Telecommunications System (UMTS) Core Network (CN) architecture, as described in Release 99 (R99) specification by the 3G Partnership Program (3GPP) [3]. Its architectural view is depicted in Figure A.1 of Appendix A. The reason for choosing the R99 as the representative packet-switched architecture of the UMTS CN lays on the deployment of the 3rd generation telecommunication network, whereby a migration from the currently used Global System for Mobile (GSM) communications [4], [5] should be supported.

In particular, the work entails the specification and implementation details of a protocol stack architecture. This has been named the Future Core Network System (FCNS) and has been developed to accommodate the necessary network and protocol security functions that should be used to protect the system from possible attacks. The structure designed represents a model framework that could be used under any packet-switched environment, serving as the means of primarily protecting the stack architecture against attacks aiming at the operation of the set

of communication rules and consequently providing the appropriate functions to secure the user and/or signalling data exchanged.

The motivation of this work is the disadvantages of currently used protocol architectures supporting communication networks and these are discussed in Chapter 3. FCNS provides the capability to secure all layers of the communication process, throughout the connection phases that protocols might be involved with at that time. The necessity of the application of the security functions is at the discretion of the network operator, although it is recommended that the FCNS functions be enforced and employed thoroughly, according to the protocol specification given in Chapters 4,5 and 6.

Finally it should be noted that FCNS has been designed as a framework for use in network environments, with existing protocol architectures serving as one of its layers, if that is requested by the network operator. It has been realised in software to provide a measure of its functionality and compare its performance against standardised models.

1.2 Methodology

Protocol design constitutes a challenging task due to the diversity of the systems that have to be supported and the identification of the appropriate parameters for its implementation. Reduced complexity, clear definition of the protocol functions, flexibility, portability and interoperability with other systems are just a few of the parameters that need to be taken into account to provide a system that is efficient, easy to comprehend and easy to use [6].

The approach taken for the realisation of the FCNS architecture follows the principle of specifying the functions of the protocol and its verification against the standards addressed in its design, prior to transforming it into a software or hardware entity.

The proposal has to be clearly defined and detailed in its specification, which should then be checked for logical consistency and conformance to the parameters and issues surrounding the environment for which applicability is addressed. Figure 1.1 illustrates the steps taken to complete the development of the FCNS stack architecture, following the approach of a software – based implementation for simulation purposes.

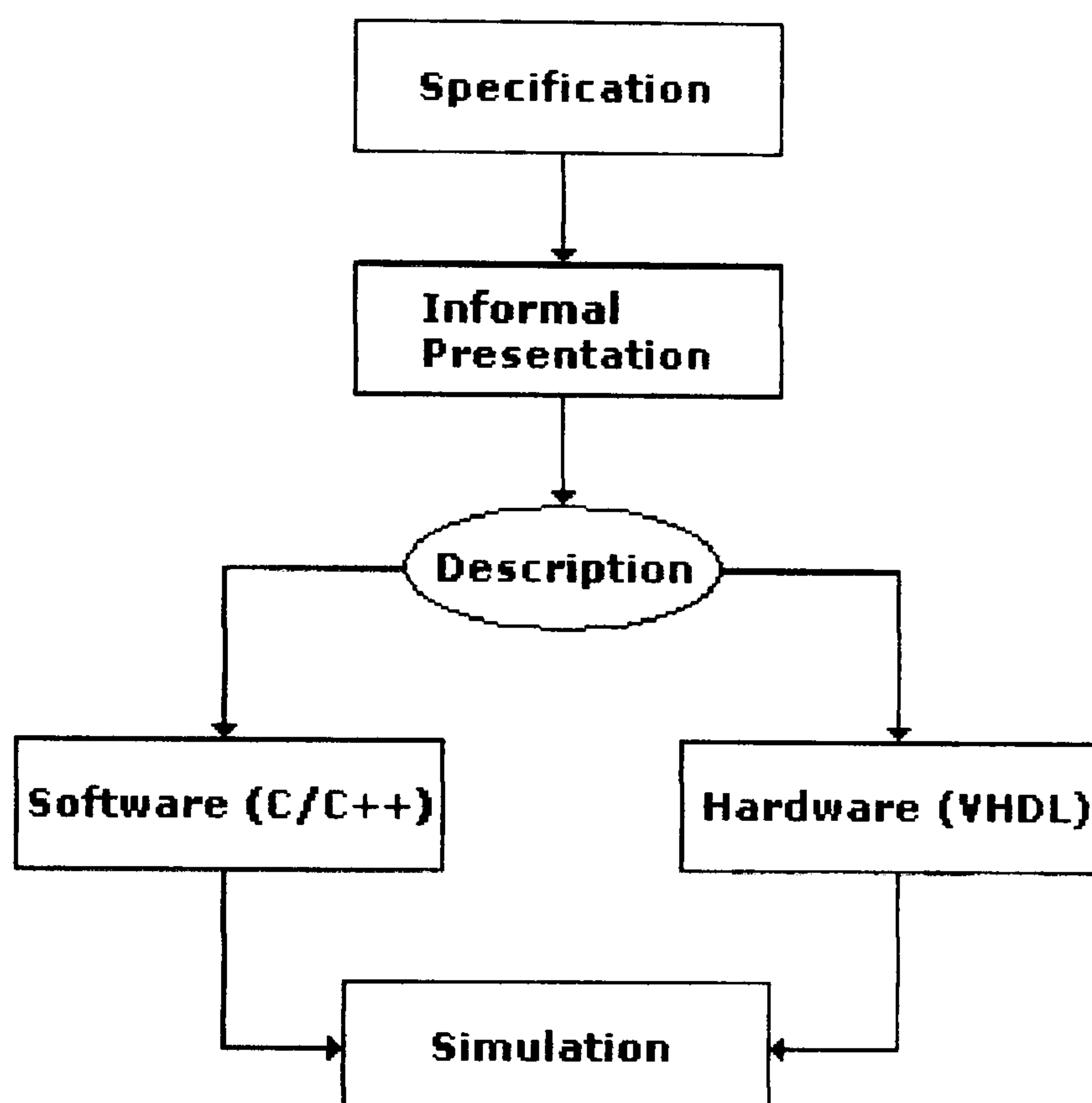


Figure 1.1: Protocol design and development steps

The specification of FCNS has been created to provide the basis upon which the rules of the protocol stack are supported, presenting the capabilities and functions of the protocols that can be used in a network architecture. It thus contains details of the messages used for realising the services provided to the system nodes, together with recommendations for the use of the FCNS functions for achieving the maximum possible performance and network resource utilisation.

To abstract the requirements addressed into a functional prototype for testing and verification purposes, the FCNS has been presented in flow chart diagrams, with respect to its layered protocols and the services offered to their users. Prototype building forms an important step in validating the conformance of the protocol to its

specification, enabling the identification of pitfalls, such as deadlocks and livelocks, where the system could enter a state halting its operation or lock into a particular state that cannot recover from. The design requirements of the protocol stack are given in Chapter 7 of the thesis, including any assumptions taken as to the network environment for which the architecture has been designed.

Usually, in high-level designs, the prototype building is followed by its interpretation using a description language, such as the Specification and Description Language (SDL) [7] or the Unified Modelling Language (UML) [8]. The approach has the advantage of providing additional means to validate the system prototype using standardised methods and techniques. Currently available tools [9], [10] support the translation of the SDL code into a higher-level language such as C++, substantially reducing the work involved in a source code implementation of the stack model.

As a final step in completing the design of the proposed architecture, the protocol functions and states have been translated into C++ source code for use within a network simulator environment. The OMNET++ simulator [11] has been chosen for obtaining performance measurements for the FCNS structure, as described in Chapter 8, identifying the conformance of the complete stack to its specification, as well as evaluating its efficiency against numerous operational conditions.

Different approaches can be followed to obtain validation prototypes and then move into realising the model into a functional software or hardware unit. No matter the tools available though, the steps depicted in Figure 1.1 constitute the foundations for the methodological design, testing and development not only of protocol but general system architectures intended for use within real operating environments.

1.3 Original Contributions

The completion of this thesis has led to a number of contributions in the field of study, including:

- The design and development of a novel secure protocol stack architectural framework, realising security services to all levels of protocol communication in a network environment running the FCNS stack.
- A secure structure addressing the problem of protocol security for the set of communication rules itself, against possible active attacks.
- The provision of a secure error signalling protocol supporting error notification services used between internal to the stack and peer network element entities.
- The design of a keystream generator architecture utilising existing pseudorandom number generation technology and one-way hash functions. The secret keys produced constitute the parameters of the security contexts exchanged between the FCNS security and communication layers.
- The functional realisation of the proposed architecture, yielding greater performance to existing network protocol structures, such as the TCP/IP suite, validated through simulation. Each of the FCNS layers has been implemented as a separate set of communication rules affording numerous services to its user, irrespective of currently used protocol architectures.
- The investigation, simulation and analysis of the FCNS use in a generic packet-switched architecture and the UMTS core network, deducing comparisons with standardised network protocols.

1.4 Terminology

Throughout the thesis there are several terms used to facilitate the composition of the material and its presentation. Unless explicitly defined in the document, the following assumptions and expressions should stand for:

- “*Application*”: The term is used to indicate either the FCNS stack user residing on top of the User-Defined Presentation (UDPRES) layer or a network element active in a network connection using the FCNS stack protocol instances.
- “*Threat*” or security threat will denote an entity possessing some danger to an information element (*asset*).
- “*Attack*” is the active attempt of an adversary in manipulating the network, either to disclose information or ultimately cause the termination of a connection. In a sense, an attack is the realisation of a threat.
- “*Network element*” denotes any node of the network used for the realisation of its services to the users connected to it. Those users will also be referred to as network elements.
- “*Protocol*” is the set of communication rules used to provide a specific set of services throughout the communication phases.
- “*Protocol user*” is the process residing on top of the protocol making use of its services, to provide access to these, either to the layer above, or to the application running the instance of the protocol.
- “*Information*” and “*Data*” are terms closely related and therefore may appear in the thesis as signifying the same notion. It should be noted though that they differ in the sense that information is what a user really wants to transmit, whereas data is the format used to enable the transmission of the information.
- “*3G*” will be used to denote an integrated system whereby a subscriber is given connection capabilities with users belonging to diverse networks and environments.

The exact details of the mapping of a protocol's Service Data Unit (SDU) into a Protocol Data Unit (PDU) are not given, since their realisation is based on the functions of the simulation program(s) available.

1.5 Thesis Organisation

The thesis has been written in nine chapters including this one. An overview of the work presented subsequently is given below:

- **Chapter 2** constitutes an introduction to packet-switched architectures including computer and telecommunications systems technology and protocols used to support their operation.
- **Chapter 3** provides the network and protocol security issues currently addressed in deployed systems. The disadvantages of the mechanisms used are presented, since they comprise the motivation of the work presented thereafter.
- **Chapter 4** introduces the FCNS architecture and more specifically its communication layers structure, entailing details of the protocols' functions and services offered.
- **Chapter 5** moves into analysing the FCNS security layer responsible for the provision of the security functions and services afforded to the FCNS stack. A detailed description of the FCNS keystream generator is also provided.
- **Chapter 6** analyses the FCNS error protocol, outlining its architecture and functionality with respect to the FCNS layers and the peers involved in a communication process.

- **Chapter 7** introduces the details of the FCNS implementation as a valid prototype, as well as a functional software instance for use within a simulation environment.
- **Chapter 8** considers the simulation environments created for the provision of the FCNS efficiency and security implication measurements. Observations and results are therefore analysed in this chapter.
- **Chapter 9** provides the conclusions of this work, given the implementation suggestions and future work that could be undertaken to improve the applicability of the stack architecture.
- **Appendix A** includes the specification diagrams of the FCNS protocols, in the form of flow charts.
- **Appendix B** represents the results obtained for the statistical measurements of the FCNS keystream generator.
- **Appendix C** presents the list of publications to date resulting from this work.

Chapter 2: Communication Systems

2.1 Overview

2.1.1 Circuit-Switched Systems

2.1.2 Packet-Switched Systems

2.2 Wireless Telecommunication Network Systems

2.2.1 Overview

2.2.2 Supporting Protocols

2.2.2.1 Asynchronous Transfer Mode (ATM)

2.2.2.2 Stream Control Transmission Protocol (SCTP)

2.2.2.3 Internet Protocol version 6 (IPv6)

2.2.2.4 Transmission Control Protocol (TCP)

This chapter provides an introduction to data transmission based on packet message structures and packet-switched (PS) technology. An overview of the history and current network status is given, followed by discussion of telecommunication environments. More specifically, the Universal Mobile Telecommunications System (UMTS) is presented, forming the integration of the fixed and mobile network systems often referred to as the Third Generation (3G) communications network. Finally, an insight into currently used protocol architectures is given, entailing details of the communication rules supporting information exchange in the UMTS Core Network (CN).

2.1 Overview

Transmission systems technology has addressed communication needs between individuals, across distances greater than would normally be possible without the use of additional mechanisms [12]. Architectures such as the Public Switched Telephone Network (PSTN) [13] and computer networks [14] have enabled information exchange between users worldwide on a voice and/or data packet transfer basis. These networks may conveniently be categorised into Circuit-Switched (CS) and Packet-Switched (PS) architectures [13].

2.1.1 Circuit-Switched Systems

CS systems base their operation in initiating and maintaining a connection throughout the call duration. In a sense, CS networks conform to a connection-oriented transmission, allocating the appropriate network resources to the users and offering a fixed data rate. PSTN and the Integrated Services Digital Network (ISDN) [15] are the most common examples of such environments used for voice transmission connections.

Details of the switching principles supporting CS connections fall outside the scope of the thesis and will consequently not be analysed in this document. The signalling procedures of the CS environments [16], such as the common-channel technique form the very essence of a network environment where voice data is to be exchanged. The current status of such procedures is based on the application of the Signalling System 7 (SS7) protocol architecture [17], which is used in various applications ranging from the PSTN, to the UMTS core network CS domain [18]. Figure 2.1 depicts the architectural view of the SS7 protocol stack and its relation to the OSI model [19].

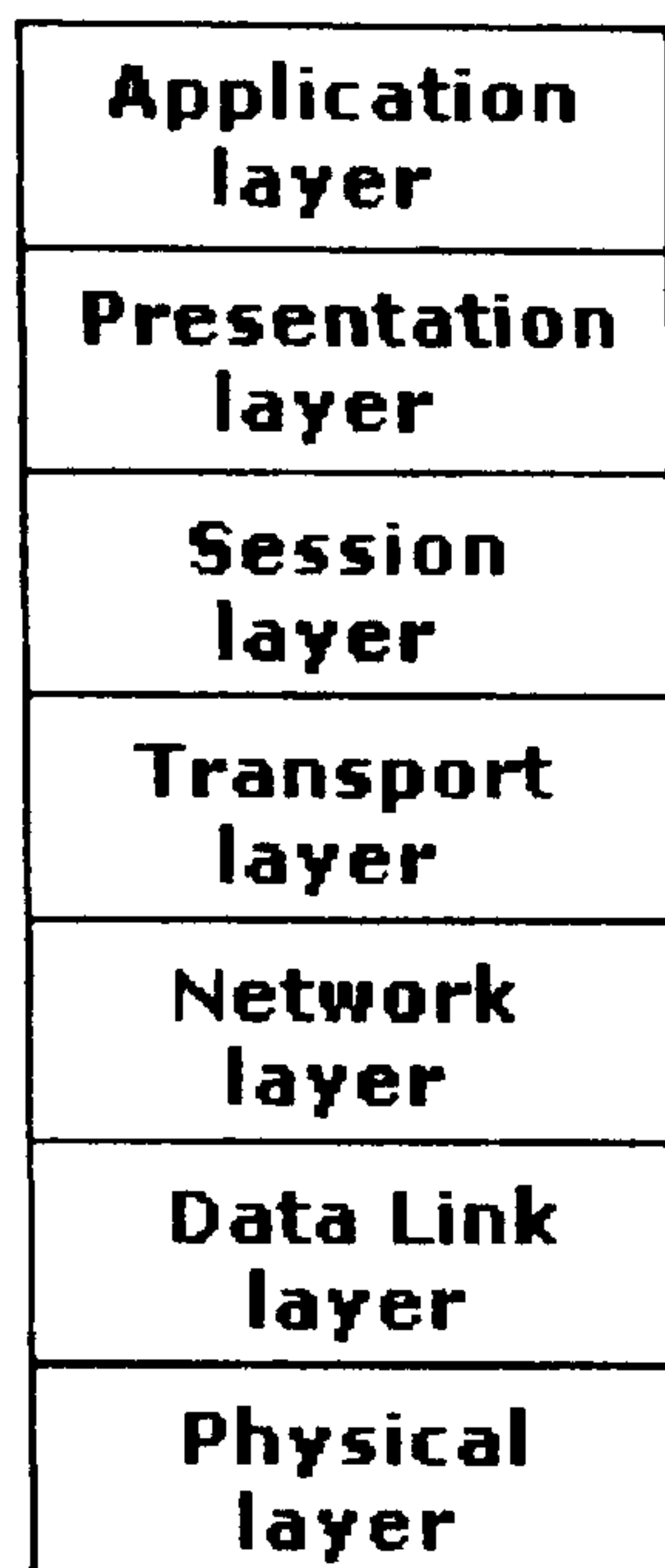
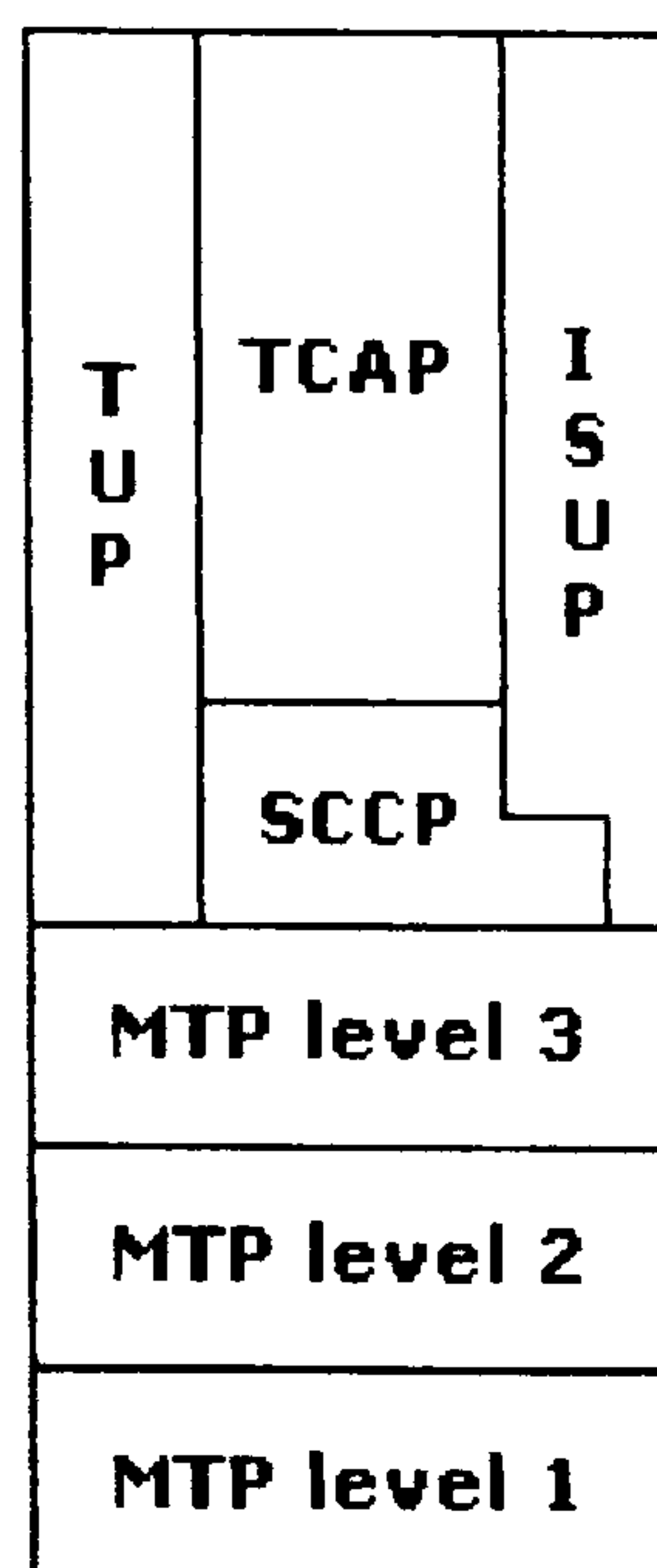
OSI 7 layer model**SS7**

Figure 2.1: SS7 architecture and its relation to OSI

The *Message Transfer Part (MTP)* is divided into three levels, each of which represents a different set of actions, defining a separate layer of the SS7 stack. *MTP level 1* encompasses the details of a physical layer protocol, including electrical and functional characteristics of the digital signalling link [20]. Data rates for the system vary from 56Kbps to 2048 Kbps depending on the physical interface used. Additionally, *MTP level 2* acts as the data link layer of the structure, being responsible for the reliable and in-sequence delivery of the SS7 messages across the signalling link. The *MTP level 3* protocol provides for the routing of the respective messages across the signalling network nodes present in the topology, and also contains functions regulating the information flow for congestion avoidance.

The *Telephone User Part (TUP)* layer has been used in some countries to serve as the means of supporting the call setup and tearing down bearer functions, though it is the *ISDN User Part (ISUP)* protocol that has replaced the TUP functions on a global scale [21]. Generally, ISUP is responsible for the maintenance and control of ISDN and non-ISDN based calls. The *Transaction Capabilities (TCAP)* layer [22] addresses issues such as the transmission of non circuit-related data using the connection provided by the *Signalling Connection Control Part (SCCP)* [23]. Both connection-oriented and connectionless services can be supported by the SCCP,

whereby an identifier is used to distinguish between the various signalling points the application instance is attached to.

SS7 has been successfully implemented in digital telephone exchanges, enabling the realisation of the ISDN services to the subscribers, offering on-demand 64Kbps rates for voice and data transmission [24]. Moreover, the reliability of the protocol suite has enabled its employment in the UMTS architecture, supporting the CS domain of the fixed network part for the voice calls made between the users [3].

2.1.2 Packet-Switched Systems

Packet-switched (PS) architectures were designed and presented in the 1960s as attempts made by research institutes to progress from CS structures to network communications using packets [25], [26]. Following the publication of these works, the first ever packet network, the ARPANET, was realised in 1969 encompassing only four nodes. By the middle 1980s, the specification of the TCP/IP protocol suite [27] enabled the growth of the primitive network, essentially renaming it to Internet and providing connectivity for various nodes independent of the ARPANET.

PS topologies encompass environments such as Local Area networks (LANs), Wide Area Networks (WANs) and other private or proprietary environments, as well as the Internet, interconnecting all networks under a single technological umbrella. The difference between these systems does not lay in the signalling technology governing their operation. Instead, architectures supporting the communication in a LAN are the same as that on another PS network and/or the Internet. Their distinction is based in the hierarchical view of the network, whereby smaller, controllable and trusted systems are connected via an untrusted and unknown environment.

Numerous differences exist between the two switching mechanisms influencing everyday communication. Figure 2.2 illustrates a sample connection timeline for the CS and PS techniques between two peer users [28].

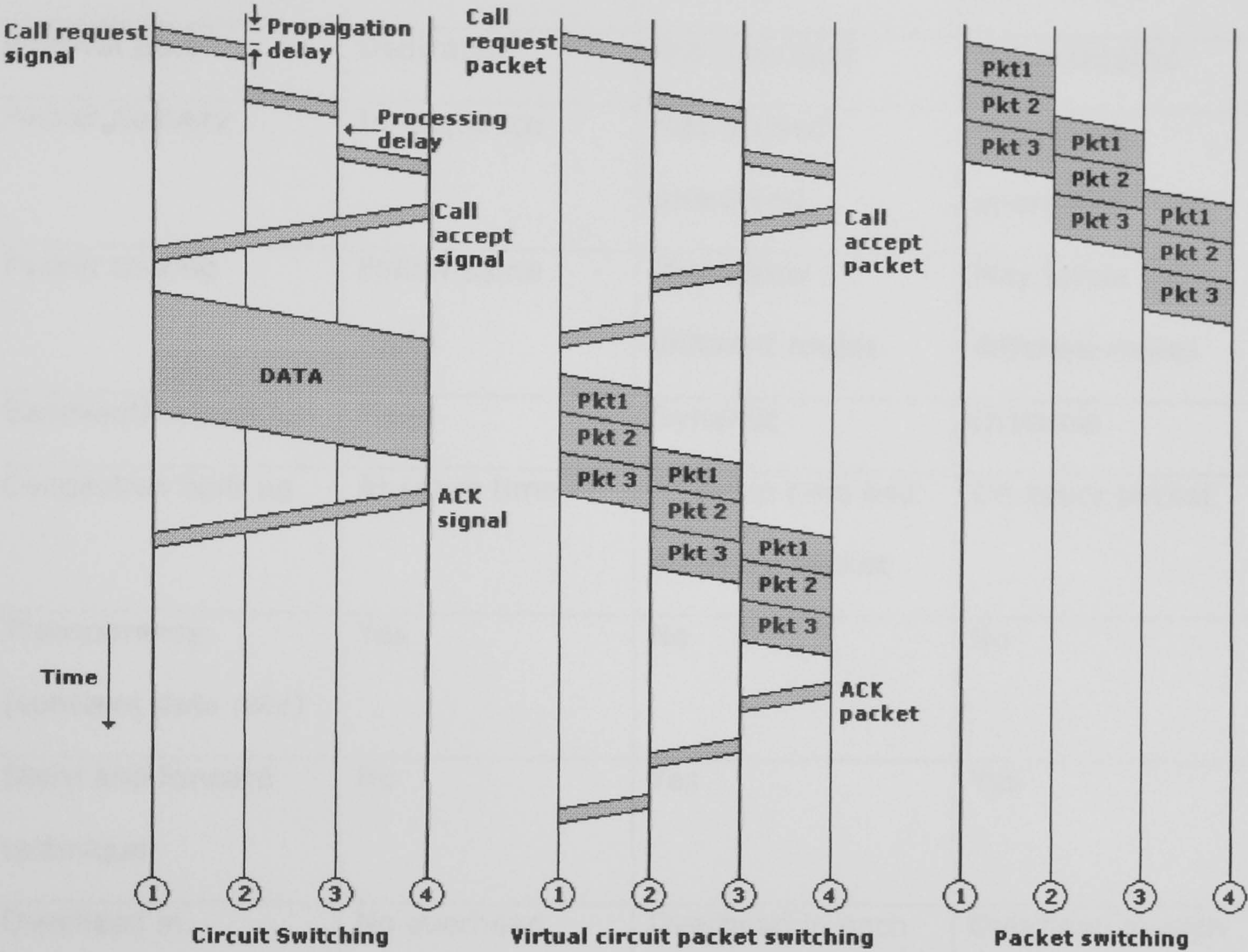


Figure 2.2: Circuit Switching vs. Packet Switching

Table 2.1 addresses the main differences between CS and PS techniques. Virtual circuit packet switching emulates the CS architecture in that a virtual circuit is setup whenever data is to be transferred. Data message transmission then follows the principles of the packet switched topologies, which are often denoted as datagram packet-switched ones.

Until the design and development of the 3G communication systems, PS and CS systems have been independent of each other [29], with only ISDN achieving a partial degree of integration. Services were offered to users via the telephone line with separate channels being used for the voice and the data messages on a CS based connection, representing a virtual circuit packet-switched mechanism.

Table 2.1: Characteristics of the CS and PS

Characteristic	Circuit Switching	Virtual Circuit Switching	Packet Switching
Call setup	Required	Required	Not needed
Physical path	Dedicated	Not dedicated	Not dedicated
Packet delivery	In-sequence	May arrived unordered	May arrive unordered
Packet routing	Follow same route	May follow different routes	May follow different routes
Bandwidth allocation	Fixed	Dynamic	Dynamic
Congestion built up	At setup time	At setup time and on every packet	On every packet
Transparency (constant data rate)	Yes	No	No
Store and forward technique	No	Yes	Yes
Overhead in messages	No overhead after call setup	Overhead in each packet	Overhead in each packet
Delay	Call setup	Call setup and packet transmission	Packet transmission

The provision of virtual circuit packet-switched connections though cannot always be achievable and there may be cases where the setup of a dedicated connection is not wanted. The identification of the appropriate switching mechanism depends on numerous factors including the size of the network, the load pattern, the host characteristics and the network topology. It is therefore up to the network operator to decide on the connection technique used and to share the system resources among multiple users and transmissions.

Connections are supported by the transmission of the information into the communication medium, either in an analogue or digital format. The technique used is strongly dependent on the technology of the underlying channel, which may vary from twisted copper cables, to optical fibres and microwave or Radio Frequency (RF) links. Digital transmission has been assumed for the FCNS implementation, with a typical system being depicted in Figure 2.3.

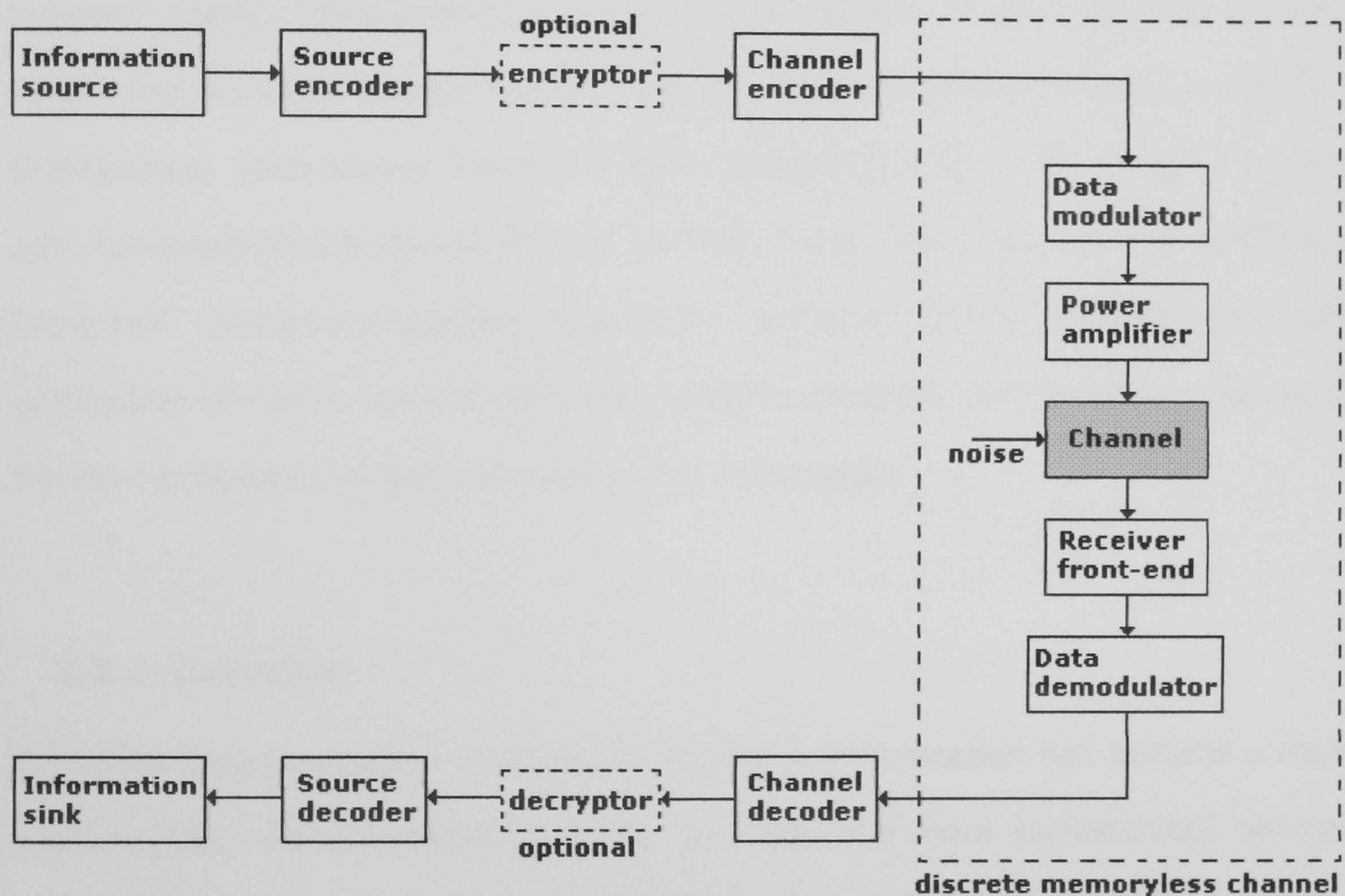


Figure 2.3: Digital communications system

Principles defining the operation of digital systems involve the representation of the required data into discrete levels and their transmission via the communication link [30]. Usually, encryption/decryption functions are implemented on a demand-only basis, since their use may not be required for all connections.

Digital communication systems have developed to provide transmission that is clearer and with less interference than the analogue based ones. Currently used architectures base their operation in faster and cheaper connections, enabling the exchange of the user information at the highest possible quality [31], [32].

Consequently, telecommunication networks, and more specifically 3G systems, have been designed to function on top of structures providing such capabilities and resources [33].

2.2 Wireless Telecommunication Network Systems

Wireless networks have been developed to support long-distance communication between users. They evolved from the initial designs of the analogue systems supporting the basic speech service, such as the Nordic Mobile Telephone (NMT) in Scandinavia, Total Access Communication System (TACS) in the United Kingdom and Advanced Mobile Phone Service (AMPS) in the USA [34]. In the 1990s the European Telecommunications Standards Institute (ETSI) adopted the new generation of mobile communications, the GSM, supplying the necessary resources for the introduction of data services to the subscribers.

2.2.1 Overview

Since the implementation of the GSM, mobile communication has become a mass market [35]. Requirements for higher bit rates and more sophisticated services leveraged research on mobile networks, leading to the design of UMTS. The need for standardisation for open interconnection systems providing full roaming services [36], have enabled the migration from the second-generation (2G) systems to an advanced network, integrating fixed and mobile technologies to supply the maximum possible Quality of Service (QoS) [37], [38]. Simultaneously, the integrated system has addressed the need of a globally accepted relocation to allow interoperability with existing network technologies [39].

The principles behind UMTS dictate that a user should be able to access the services provided no matter his/her geographical position with respect to the peer entity and the network provider. However, global connectivity issues remain on a

theoretical level, with their implementation being dependent on the interoperability between providers and the data rates that could be available to the subscribers. Figure 2.4 depicts the basic concept of communication for 3G mobile systems.

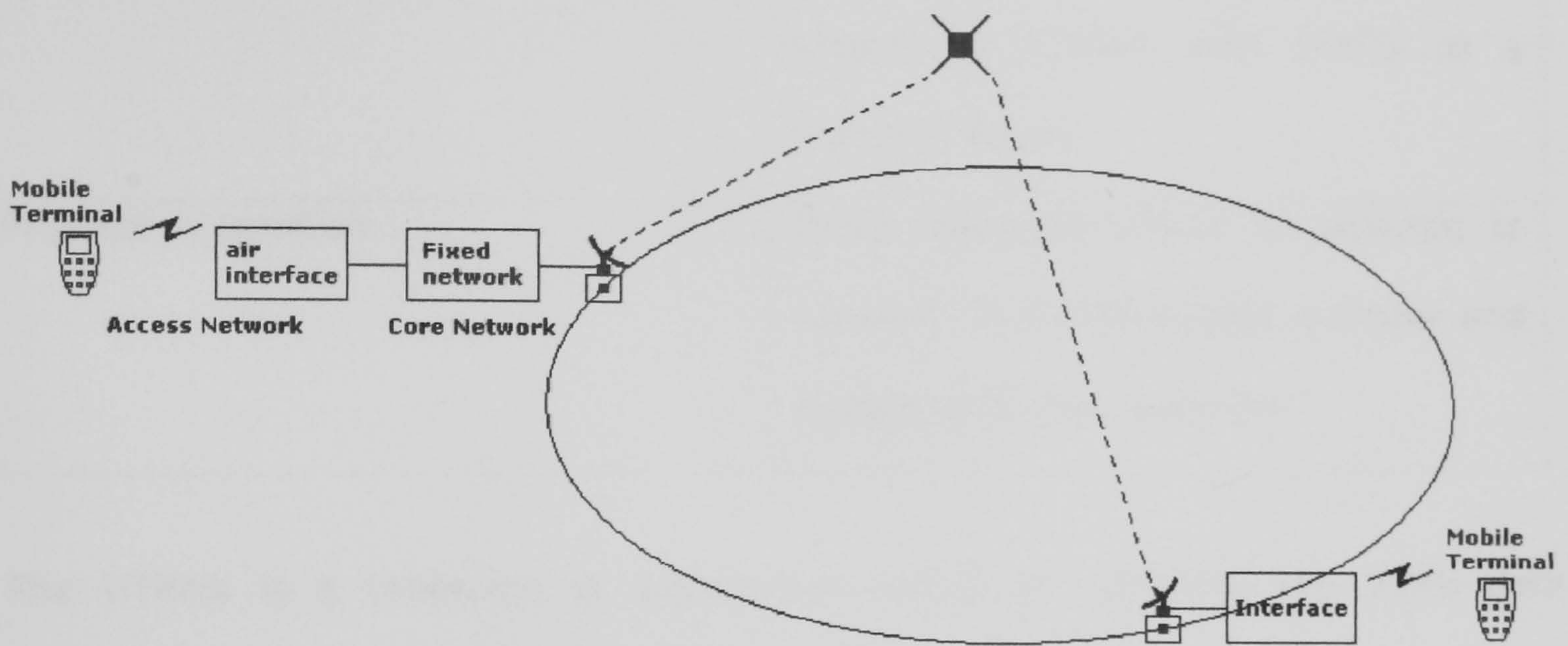


Figure 2.4: Basic UMTS connectivity concept

UMTS consists of three different network environments. Its main components are the access network [40], [41], often referred to as the UMTS Radio Access Network (UTRAN), and the fixed network part, denoted as the Core Network (CN). The satellite component of the UMTS (S-UMTS) has been designed to compensate for global connectivity issues between users belonging to different physical locations.

This system integration aims at optimally using the common networks that may be present and avoiding any unnecessary functional duplication. Table 2.2 identifies the main areas migration towards UMTS targets [42].

Table 2.2: UMTS integration domains

Integration setting	Aim
Service integration	Services offered to the user are the same for both fixed and mobile communications

Infrastructure integration	Transport and switching infrastructure between UMTS and B-ISDN communication should be shared
Functional integration	Functional entities in the fixed network should be shared with UMTS to a maximal extend
Protocol integration	Fixed protocols should be selected to support UMTS information transfer and signalling at the interfaces

The UTRAN is a collection of subsystems aimed at switching the voice data information between the subscribers, identifying the necessary parameters for the setup of the call connection [43]. It is the cellular environment of UMTS adopting a mixed cell architecture basing its operation in architectures such as the Wideband Code Division Multiple Access (W-CDMA) [44 - 47] and Spread Spectrum techniques [48 - 51].

S-UMTS is a system still under development, with its implementation awaiting the commercial deployment of the UMTS and the interoperability between the various service providers worldwide [52].

Finally, the UMTS CN is a collection of network elements serving as the main switching centre of the integrated network, for both voice and non-voice based data. Its architecture has undergone several modifications varying from the Release 99 (R99), to the Release 6 (R6) specifications [53], with emphasis being given to the underlying protocols supporting its operation [54], [55]. The CN consists of two domains, namely the circuit switched and the packet switched ones. The former is responsible for the transmission of the voice data between the peers [56 - 59], whilst the latter addresses issues regarding packet data transmission and interconnectivity with networks such as the Internet and/or LANs.

2.2.2 Supporting Protocols

The identification of advanced radio and packet services [60 - 62] led to the design of the R6 specification, which adopted an all-IP infrastructure [63], [64] to enable compatibility with currently used computer network protocol systems.

For the R99 specification there exist three different network protocol system layers, defining sets of communication rules for the exchange of user and control data [65]. The *Transport Network* architecture encompasses the protocols responsible for the reliable transmission of the user and control data across peers and the CN elements. The *Radio Network* layer is responsible for maintaining the transmission of the radio access parameters between the user equipment and the CN. Finally, the *System Network* layer has the task of monitoring and preserving the communication path set by the radio network layer entities.

All three layered architectures address functionality issues for both the UTRAN and the CN, as well as for the CS and PS domains of the core network. Their implementation facilitates the management of the mobility and session services offered to the UMTS users. The communication rules forming the stack structures are based on protocols used in computer network systems, to enable interoperability with existing system entities as well as to remove the proprietary nature of SS7 based networks. These protocols vary from the Asynchronous Transfer Mode (ATM) [15] to the Stream Control Transmission Protocol (SCTP) [66], the Internet Protocol (IP) [67], Transmission Control Protocol (TCP) [68] and the Synchronous Digital Hierarchy (SDH) [69].

2.2.2.1 Asynchronous Transfer Mode (ATM)

The ATM is a protocol designed to compensate for variable bit rate connections and enable user data transfer at very high speeds. It has initially been intended as a technique for use within the Broadband – ISDN environment to support

transmission rates of up to 155.5MBps and 622.08Mbps [70]. Currently, ATM is widely used in large corporate networks supporting their backbone operation, such as university intranets and other proprietary environments. Its functionality is based on the reference model depicted in Figure 2.5.

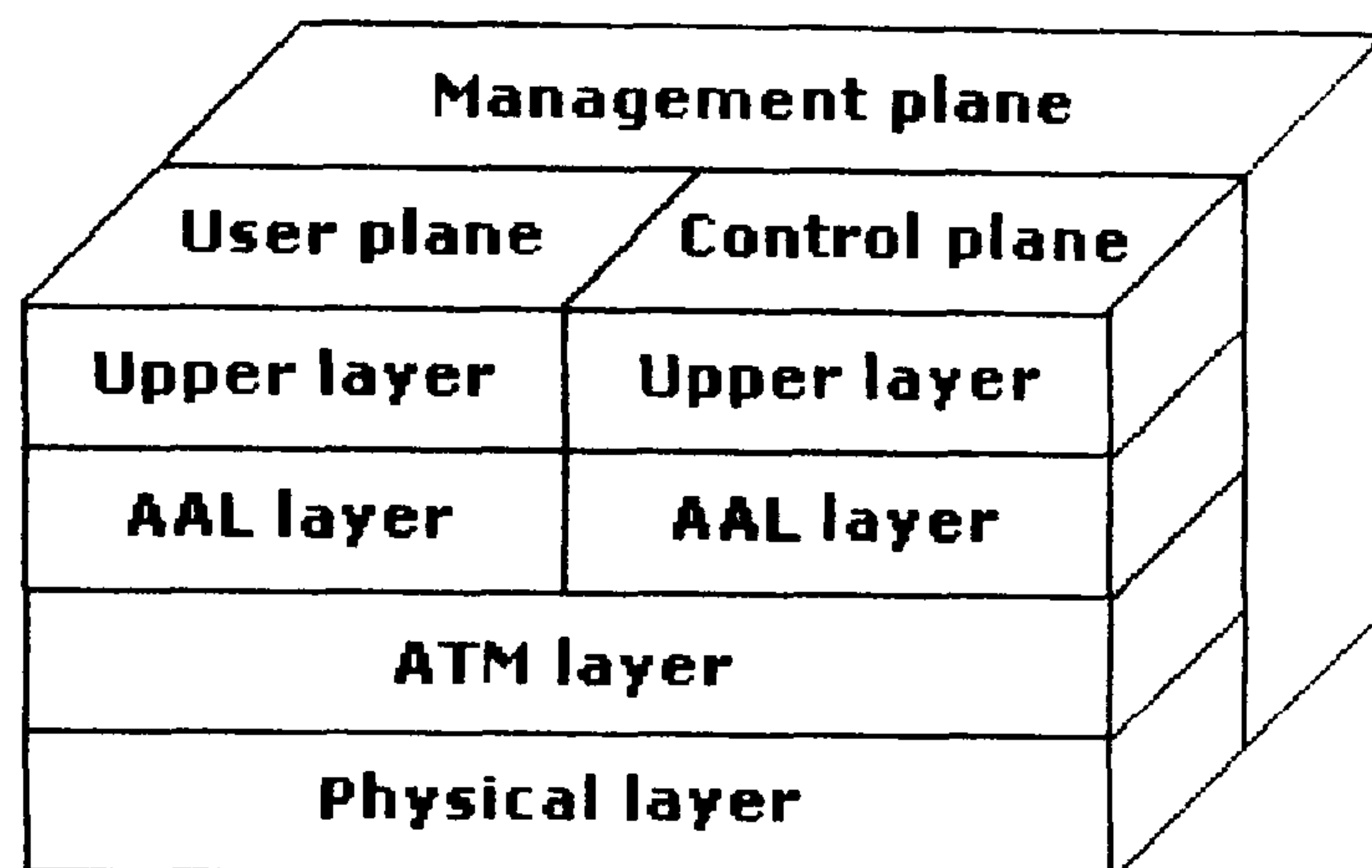


Figure 2.5: ATM reference model

In the ATM model, data exchange has been classified into two categories. Signalling information exchanged between network elements handled by the *Control Plane* and user data managed by the *User Plane*. The *Management Plane* of the reference model maintains network operability and is responsible for regulating network resource issues via the Operation and Maintenance (OAM) cells. The *ATM Adaptation Layer (AAL)* receives the data from the application (*Upper Layer*) and segments them into the 48-byte blocks forming the ATM cell payload area. Upon reception of the data to be exchanged, the *ATM layer* adds the 5-byte header to compose the ATM cell structure, prior to forwarding it into the *Physical Layer* for transmission in the communications medium. The physical layer is also responsible for the calculation of a checksum on the ATM header for error detection purposes.

ATM provides a connection-oriented service on two main levels, namely the User-to-Network Interface (UNI) and the Network-to-Network one (NNI). Their differences are reflected in the ATM cell structure, with usually the cell carrying signalling and/or control information. Parameters concerning services offered to the communicating parties are included in the data payload area, addressing

functionality issues of the B-ISDN reference model and the ATM architectural layers.

Security issues of the ATM are discussed in Chapter 3, outlining details on the implementation of the relevant framework in network topologies and the UMTS CN.

2.2.2.2 Stream Control Transmission Protocol (SCTP)

SCTP has been developed to replace the TCP stack, in cases where the use of the latter protocol is inadequate, such as the transfer of network signalling information. The reasons for this approach include the fact that some applications may not require the sequence maintenance provided by TCP, whilst others might find that the stream orientation is inefficient for their messages. SCTP has been designed in such a way that these problems would be eliminated and offer the applications more control over the packets sent throughout the network.

The particularities of the nature of the signalling data also render TCP use inadequate for numerous reasons. TCP has been designed to support the transmission of large data amounts between two end-points, implementing flow control based on the behaviour of the end-to-end traffic [71]. Signalling data sizes are comparatively smaller than those of the user data, with their traffic pattern consisting of message bursts rather than forming a sequenced stream. Typical values range from 200 bytes to 500 bytes of data, in contrast to the 1500 – byte Maximum Transfer Unit (MTU) set for the user information [14]. Moreover, security pitfalls identified for the TCP protocol (Chapter 3) affected the need for a more secured solution, offering protection against replay and sequence number attacks.

SCTP bases its operation on the messages depicted in Figure 2.6. The *source* and *destination port numbers* identify the respective ports for which the application data refers to. The *verification tag* field is used to validate the sender of the SCTP

datagram in an attempt to overcome fabrication attacks and ensure messages' confidentiality.

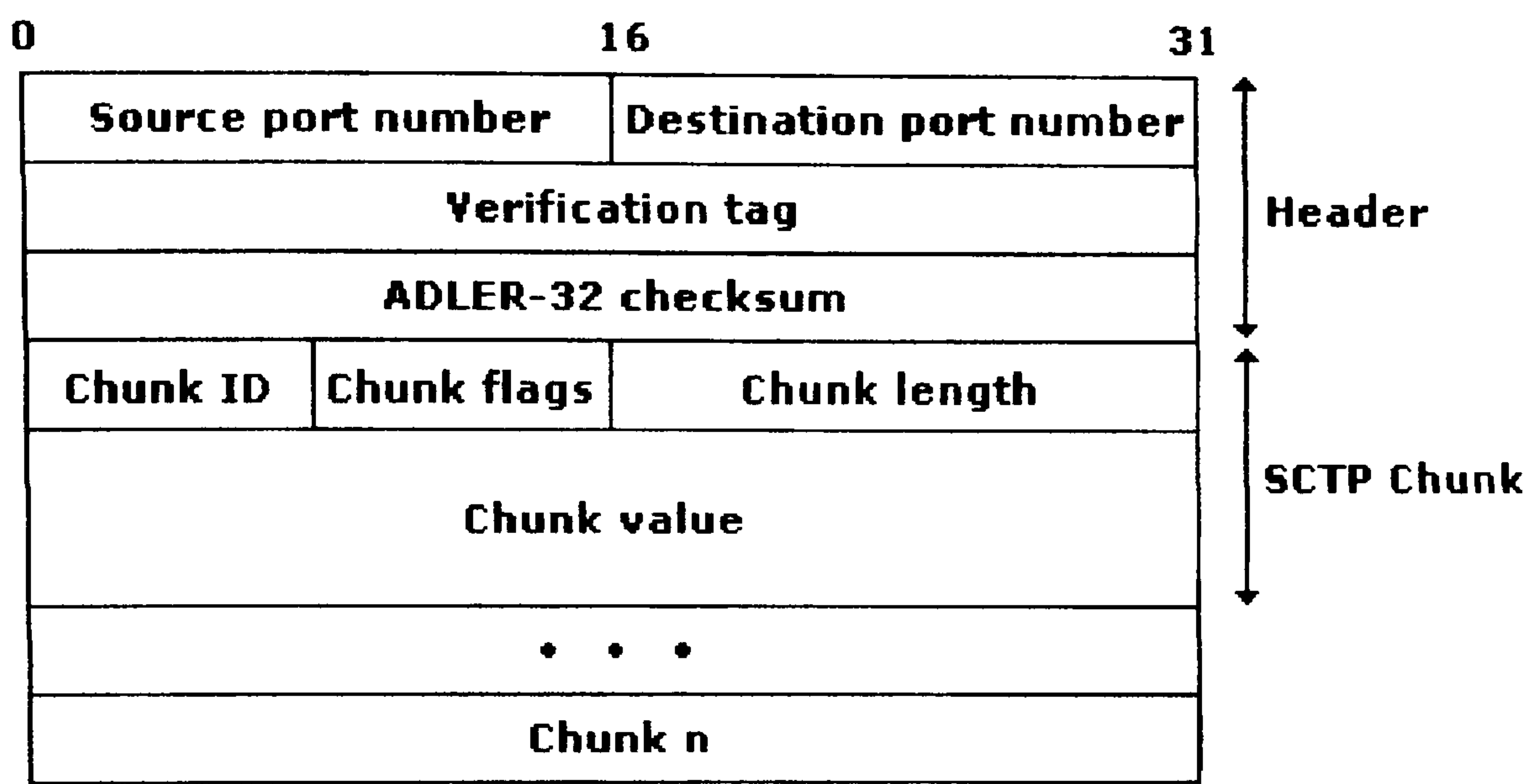


Figure 2.6: Sctp message format

During transmission, the value of this tag must be set to the value of the *Initiate Tag* received from the peer endpoint during the connection initialisation. For those data chunks carrying the initiate tag, the verification tag is set to 0, so that the receiver will be able to distinguish them between the datagrams received. Upon reception of a data block, the receiver checks the *Chunk ID* field and according to predetermined criteria decides upon the acceptance or rejection of the data. The *Adler-32 checksum* is a 32-bit field used to verify the message transmitted and identify any possible modifications. The *chunk ID* 8-bit field is used to determine the type of data contained in the *Chunk value* field. Finally, the *Chunk flag* field could be used to specifically locate individual messages inside the data stream, though it is not currently used, as described in the Sctp specification.

The employment of the Sctp protocol instead of the TCP architecture has been set to balance the end-to-end secured communication provision. Its realisation in the UMTS CN has been based on the transportation of the Session Initiation Protocol (SIP) messages, responsible for the establishment of IP specific user calls. This implies that the deployment of Sctp has been set for Release 4 and above of the

CN and not for the R99, which does not entail any mechanisms for the information protection between the various network elements.

2.2.2.3 Internet Protocol (IPv6)

The IPv6 has been designed to overcome problems associated with IPv4 concerning security issues and unreliability in the data transmission process. A wide range of services can be offered by the protocol instance, in the form of the extension headers made available by the network operator for the specific topology. Figure 2.7 illustrates the architecture of the IPv6 packet header.

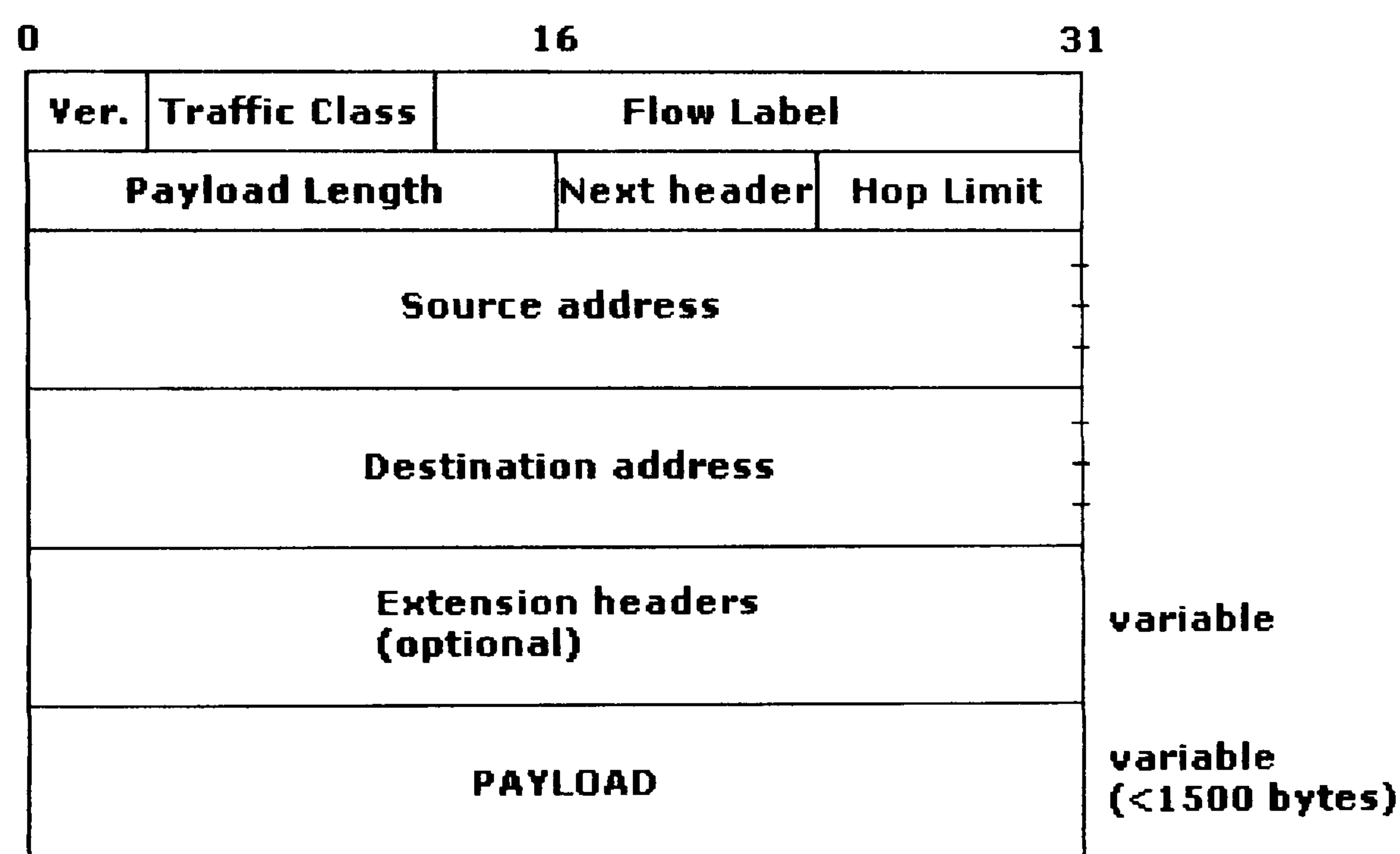


Figure 2.7: IPv6 packet header

The *version* field indicates the version of the IP stack, whilst the *Traffic class* one is used by the switches or routers to indicate the priority level that message should be given. *Flow label* is used to define the flow or the sequence of packets for which the source demands special handling from the intermediate routers. The *Payload length* field indicates the packet size, excluding the IPv6 header size. If it is set to zero, then the "Jumbo payload" option can be used, which essentially allows packets exceeding 65535 bytes in length to be transferred. The *Next header* field indicates whether any of the optional extension headers will be used for the connection, or it is the header of a higher layer protocol, such as the TCP. Finally,

the *Hop limit* field identifies the number of nodes a message could traverse before it is discarded by the network elements.

The main improvements of IPv6 compared to IPv4 are provided by its extension headers. Table 2.3 illustrates the meaning of the additional headers in the recommended order they should appear [67].

Table 2.3: IPv6 extension headers

Extension headers	Meaning (if header is present)
Hop-by-hop options header	This header is processed by all intervening routers and nodes and hence must appear first. It is used to indicate the Jumbo payload option, which can either be changed en route or not.
Destination options header	A set of options for the destination address in the packet (that destination may not always be the ultimate one, and hence the presence of a header similar to this further down).
Routing header	Contains a list of all intermediate addresses. It specifies the type of routing and source routing.
Fragment header	In IPv6 only the source node is allowed to fragment and this field specifies whether that action has taken place or not.
Authentication header	Provides data origin authentication, connectionless integrity of datagrams and anti-replay service.
Encapsulating security payload header	Provides message integrity, confidentiality and/or authentication
Destination options header	Similar to the one described above. Used if the destination for the second extension header is

	not the ultimate for the packet.
Upper layer header	Indicates the upper layer protocol used on top of IP.

The Authentication Header (AH) and the Encapsulating Security Payload header (ESP) form the basis of the IP security architecture (IPsec). Their use enables the system to afford the required security features in the user data packets and decide upon the acceptance of the messages depending on the policy governing that connection. IPsec is analysed in Chapter 3, where an indication is given as to the drawbacks and potential implementation pitfalls of the architecture.

2.2.2.4 Transmission Control Protocol (TCP)

The TCP is used as the means of ensuring the error-free and efficient transport of the user information towards the intended destination. It is an end-to-end connection mechanism providing a connection-oriented service to the upper layers, managing the flow of datagrams from those layers towards the IP layer and vice versa. Upon transmission of the packets, TCP uses the connection (the virtual circuit) as its fundamental element. By this method, a protocol port can be utilised for several connections at the same time, depending on the application entity running on top of the stack. Figure 2.8 illustrates the structure of the TCP datagram as set by its specification [68].

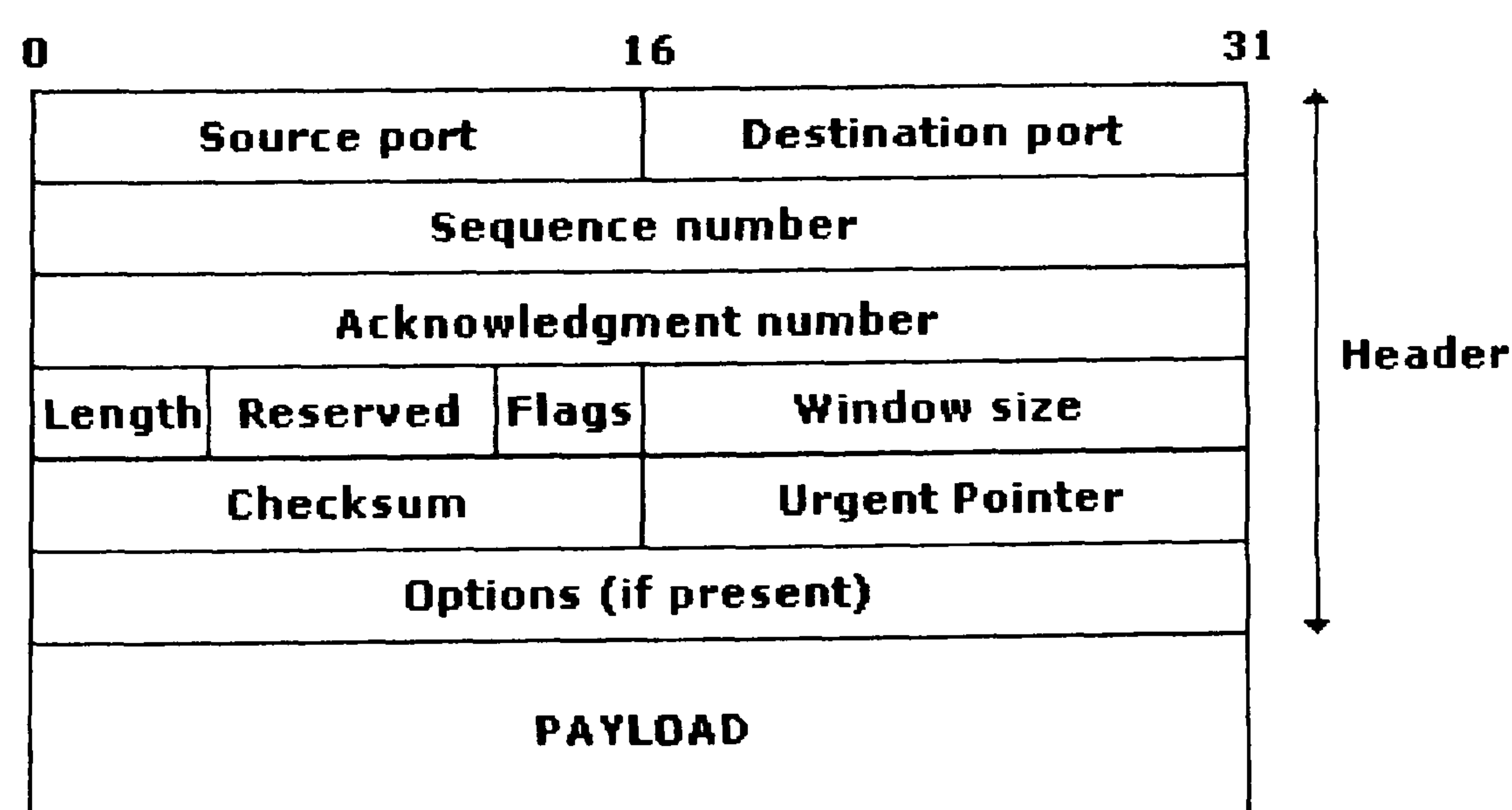


Figure 2.8: TCP message structure

The *Sequence number* field is a 32-bit number indicating the current block's position with respect to the overall message. It is also used to provide the initial send sequence number to the receiver. The *Acknowledgement number* field is the number indicating the next sequence number expected. If the TCP message is sent back to the sender, then the field is used to indicate the sequence number of the last data received, which will be the last sequence number received plus one. The *Length* or *data offset* field indicates the length of the TCP header in integral multiples of 32-bit words. *Reserved bits* are set to 0, since they are never used and hence ignored by the communicating entities. The *Flags* have various meanings, indicating necessary actions to be taken for each connection. Table 2.4 summarises their uses in the TCP stack.

Table 2.4: TCP flag field meanings

Flags	Meaning
URG flag	Urgent pointer flag. If it is set it indicates that the urgent pointer field is significant.
ACK flag	Acknowledgement flag. If it is set, it indicates that the acknowledgement field is valid.
PSH flag	If it is set it means that the <i>push</i> function is to be performed.
RST flag	Reset flag. If it is set, then the connection must be reset.
SYN flag	Sequence number flag. If it is set, the sequence numbers are to be synchronised between peers. This flag is mainly used during the connection establishment process.
FIN flag	If it is on, then the sender indicates that it has no more packets to send.

Furthermore, the *Window size* field is a number indicating the number of blocks of data that the receiving machine can accept. The *Checksum* is calculated by taking the 16-bit one's complement of the one's complement sum of the 16-bit words in

the header. The *Urgent Pointer* field indicates the portion (blocks) of data that should accordingly be treated. The action that will be followed though is determined by the application and not the TCP stack itself. Moreover, there are currently only three *Options* defined for TCP. One indicates the end of option list, whilst the second one indicates a no operation status and the final option the maximum segment size.

SDH and protocol architectures such as the GPRS stack family are used to compensate for UMTS specific issues, including the digital transmission of the user and control data, as well as the routing of the respective information within the UMTS CN elements. Their implementation though is subject to proprietary and operator standards and hence will not be dealt with in this work.

The CN R99 PS domain has been taken into consideration for the completion of this thesis, since the initial UMTS commercial applications will be based on that architecture. Its advantages and implementation pitfalls, as well as the drawbacks with respect to security of the R6 are also described in Chapter 3, forming one of the main motivation notions behind the design of FCNS.

Chapter 3: Network Protocol Security

Chapter 3. Network Protocol Security

3.1 Overview

3.2 Network Attacks

3.3 Protocol Security

3.3.1 Architectural Analysis

3.3.1.1 ATM Security Architecture

3.3.1.2 SCTP Security

3.3.1.3 IP Security Architecture (IPsec)

3.3.2 OSI Security Model

An overview of current network and protocol security systems is given, including an analysis of the type of attacks that could be launched against the FCNS and the environment in which it may run. The architectures currently supporting the secured operation of computer and telecommunication systems are also provided. The disadvantages of the structures analysed are given with respect to the implications their employment may have. Furthermore, an analysis of protocol stack security architectures is presented, concentrating on the OSI security architecture. The disadvantages of this model are further provided together with a comparison with the FCNS stack architecture. The drawbacks and implementation pitfalls of the solutions analysed form the motivation behind this work.

3.1 Overview

This chapter offers an insight into the security of communication protocols supporting connections in packet-switched architectures. For the reasons identified in Chapter 2, their use has been extended in enabling communication establishment and maintenance in the UMTS CN PS domain. The analysis provided will focus on the latter environment, where standardisation bodies have addressed requirements in addition to those present for computer network architectures [72], [73].

The identification of security threats and countermeasures for telecommunication systems resulted in the development of the Network Domain Security (NDS) architecture of the UMTS CN. The NDS mechanisms though have been omitted from the R99, leaving the network vulnerable to both passive and active attacks [74 - 76]. In contrast, R4/5 of the UMTS CN adopted an all-IP infrastructure, where IPsec [77] and the Mobile Application Part (MAP) security architecture (MAPSEC) [78] have been proposed to protect the user and signalling data in the PS and CS domains respectively.

Computer network security technology has historically addressed only the need for protecting the user and/or signalling data transit in a given topology. Encryption mechanisms and hash algorithms are continuously being developed to enforce integrity and authentication functions to the information exchanged [79 - 81]. These solutions lack the specification of mechanisms that could be used to protect protocol stack layers from active attacks and to ensure the integrity of the interlayer messages sent between the layer protocols of the structure [82]. The efforts of the International Standardisation Body (ISO) to provide a secure stack framework led to the development of the OSI security architecture [83], [84]. The work, however, represented efforts in placing security functions inside the OSI model layers, to enable their management in providing protection only to messages external to the operation of a protocol [85].

An approach ensuring the security of the internal stack messages has yet to appear in current protocol specifications. FCNS has been designed to address the issues concerned with the protection of the service primitive messages used to establish communication parameters between the stack layers. Additional mechanisms could then be used to afford the required functions for the security of the signalling and/or user data exchanged in the topology.

3.2 Network Attacks

The UMTS model, where computer and mobile systems have been merged, implies the use of the same set of communication rules for supporting both environments. This protocol simplification has influenced the design of telecommunication systems, since an attacker would now be able to launch attacks similar to those in computer networks.

Communication channels are frequently susceptible to potential attacks or threats from malicious users, aiming at either obtaining information as to the data transferred, or modifying and fabricating that data at will. Both computer and mobile systems are subject to unauthorised connection and data requests, with adversaries intending to use services and resources offered by the networks mainly for causing Denial of Service (DoS) attacks against legitimate users.

Security in the FCNS environment has been focused on packet-switched architectures, such as computer networks [14] and the UMTS core network (CN) Release 99 [3]. The security threats presented in this section have been described on the packet-switched network basis, meaning that issues concerning radio access networks have been left outside this discussion. Work has already been completed for the access system of the UMTS, where security functions are afforded to secure the radio links that messages traverse [86].

Regardless of the means used for attacking a system, network attacks can be categorised according to the intentions of the adversary manipulating the network. Figure 3.1 illustrates this classification, common to both mobile and computer environments [28].

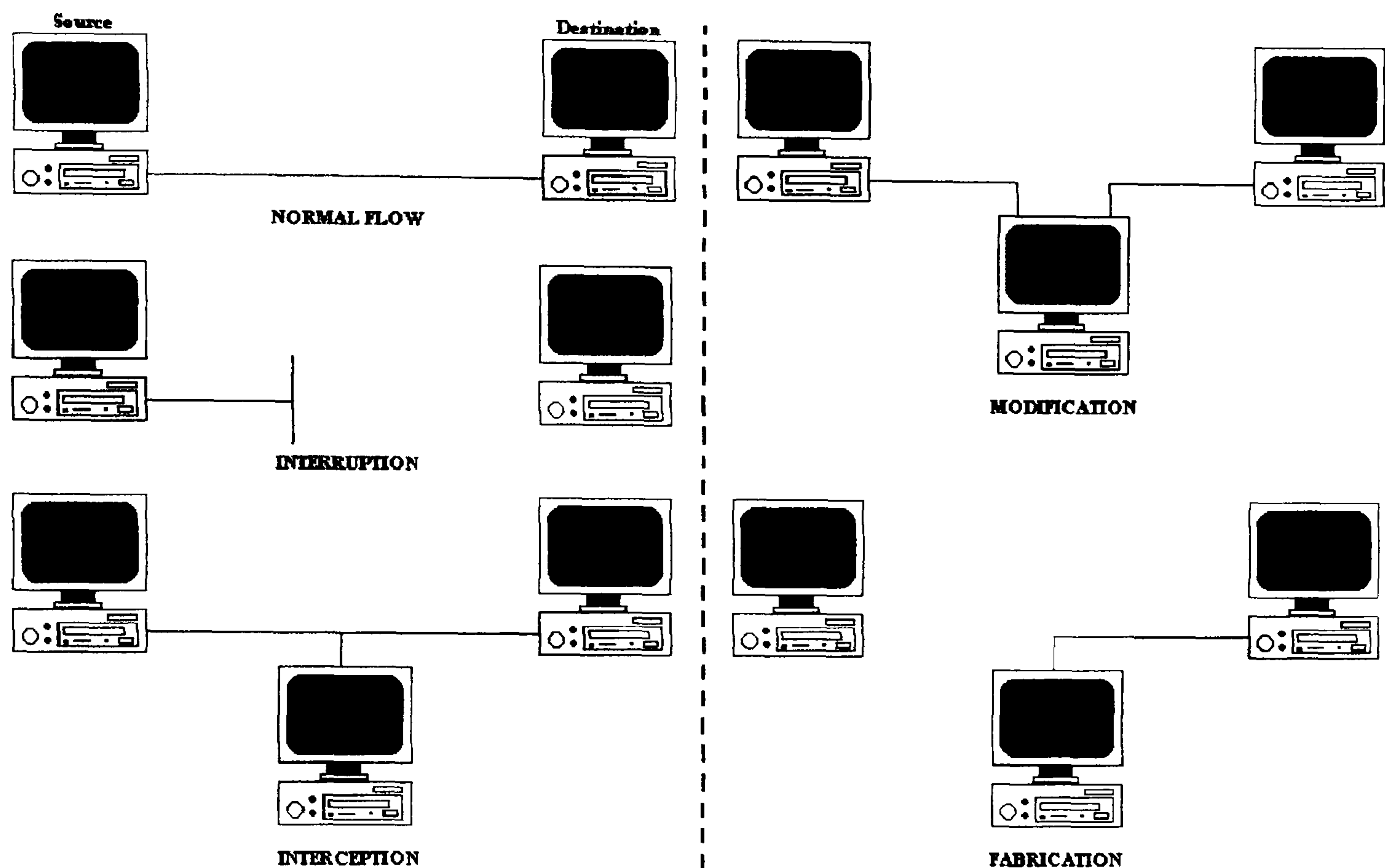


Figure 3.1: Classification of telecommunication network attacks

Interruption is an attack targeting the availability of a system, where a service or system property becomes destroyed or unavailable, or even unusable. Typical examples include hardware destruction, communication line disconnection and other attacks that can be mounted on the physical device supporting the communication.

Interception attacks aim at disabling or overcoming the confidentiality features of the communication, whereby an attacker gains unauthorised access to a system resource or data message, by either wiretapping or using packet sniffing tools. Usually, interception attacks do not address connection discontinuity issues, unless the adversary moves into launching an interruption attack.

Modification threats result in attacks against the integrity of the user and/or signalling data, with the adversary aiming at gaining unauthorised access to a service or tampering with the data. It is an attack that may occur in a network environment, where no encryption mechanisms have been used to secure the network resources and/or user data in transit.

Finally, *fabrication* or *fraudulence* attacks are categories whereby message authentication is targeted by an illicit node counterfeiting objects into a system. Examples include the insertion of malicious packets in the network such as viruses or Trojan horses [87]. Usually this type of attack comes as result of a successful node impersonation following an interception attack, where users do not authenticate their data. Additionally, bogus packets could be inserted into the data flow, to increase a node's processing load, build up network congestion, or even occupy network resources in such a way that the node will become unable to accept any requests made by legitimate nodes.

Further consideration of the threats to telecommunication systems, enables them to be additionally classified into two main categories [28].

Passive attacks, in which transmission is monitored or eavesdropped to obtain information as to the data being exchanged. Such attempts are usually launched against vulnerable communication media, such as coaxial cables, where wiretapping can be possible. There are several types of passive attacks, including:

- Release-of-message contents, which implies the disclosure of the information sent between the peers
- Traffic analysis, targeting systems providing for message integrity mechanisms, to exploit the location and identification of the communication parties. A consequent attack, namely traffic pattern analysis, aims at observing the frequency and length of the messages sent to guess the nature of the communication.

Passive attacks are usually difficult to detect, since no data alteration is involved leaving the data stream intact for the peers. Security measures have been developed to prevent such a case, by mechanisms varying from authentication and data encryption, to traffic padding techniques for the traffic pattern analysis avoidance.

Active network attacks, which involve the modification of the data in transit or the creation of false messages and connection requests. These include:

- Masquerade attacks falling into the category of fabrication attacks, where the adversary impersonates a valid communication node, to either establish connection requests or capture data packets and cause interruption attacks.
- Replay attacks involving the replay of a part or a whole communication session, having obtained information as to the identities of the peer entities and the type of data that has been transferred in the captured session.
- Modification attacks, which an adversary launches to alter a portion or the whole of the data message or stream to produce the desired effects.
- Denial of Service (DoS) attacks targeting the connection lifetime and management, whereby the adversary tries to cause the denial or prevention of the service access to the legitimate users, leading to connection releases and buffer overflows. A possible extension of this type is the Distributed DoS (DDoS) attack, where many attackers are joined together in an attempt to quickly launch the attack against a node or a system.
- Finally, active attacks can be initiated in the form of malicious code insertion on the data stream, such as viruses and/or backdoor programs, aiming at causing connection release and protocol procedure destruction, or simply identifying any implementation pitfalls of the network architecture, which an attacker could use for launching a DoS attack.

Active attacks can also be difficult to correct or recover from, yet the proper application of the security features of the protocols used could ensure their

detection and possible recovery. Implementation of both attack categories can be found at [87 - 89].

3.3 Protocol Security

Network protocol security addresses the protection of information against unauthorised disclosure and/or theft in communication systems. The attacks presented in the previous section aim at manipulating the network in the following ways:

- Information leakage
- Integrity violation
- Denial of Service and
- Illegitimate use

There exist numerous methods of protecting a system from unauthorised access and usage of resources. Eight protection mechanisms are defined in [84]:

- *Encipherment*, to provide via encryption data and traffic flow confidentiality. Encryption can also be used to support integrity requirements.
- *Digital signatures*, to account for non-repudiation and authentication issues. Can also be used as a means of ensuring data integrity.
- *Access control mechanisms*, whereby users obtain certain privileges for accessing network resources and/or data. Typically, access control is governed by the use of lists and policies at the server side.
- *Data integrity mechanisms*, to provide protection against data modifications. These can be based on a single message or on a stream basis, depending on the service provided by the protocol supporting the connection.
- *Authentication exchange mechanisms*, whereby the communication entities exchange information concerning the processing and identification of the data.

- *Traffic padding*, which supports the flooding of the network links with random data, aiming at defying traffic analysis attacks.
- *Routing control*, to prevent sensitive data from traversing insecure channels.
- *Notarisation*, whereby a trusted third party may act as a legal representative to guarantee the data integrity, peer authentication and non-repudiation services.

Numerous other techniques can be provided that may not be specific to a security service (pervasive techniques). Only those involved with the mechanisms offered by the protocols supporting the operation of the network will be considered in this Chapter. Techniques such as firewalls [90] and managing Information Technology (IT) systems via security risk analysis and audits [91], [92] will not be dealt with, as they fall outside the scope of this work.

3.3.1 Architectural Analysis

In this section, an analysis of several modern security protocol architectures is given. Information is provided as to the mechanisms used to support the required services for the communicating peers. The protocols are also analysed with respect to the requirements set for the UMTS environment and more specifically the CN.

3.3.1.1 ATM Security Architecture

The ATM Security Architecture (ATMsec) is defined in [93] and [94], with information being given on the Network-to-Network Interface (NNI) signalling security in [95].

The identification of the security requirements for ATM environments follows the principles identified by [84]. Since transmission in ATM networks is based on packets (cells), attacks fall into the passive and active categories aiming at

manipulating the data flow. The ATMsec identifies the security Information Elements (IEs) that can be used to provide the required protection functions for the communicating peers. Figure 3.2 depicts the structure of the ATM security IE.

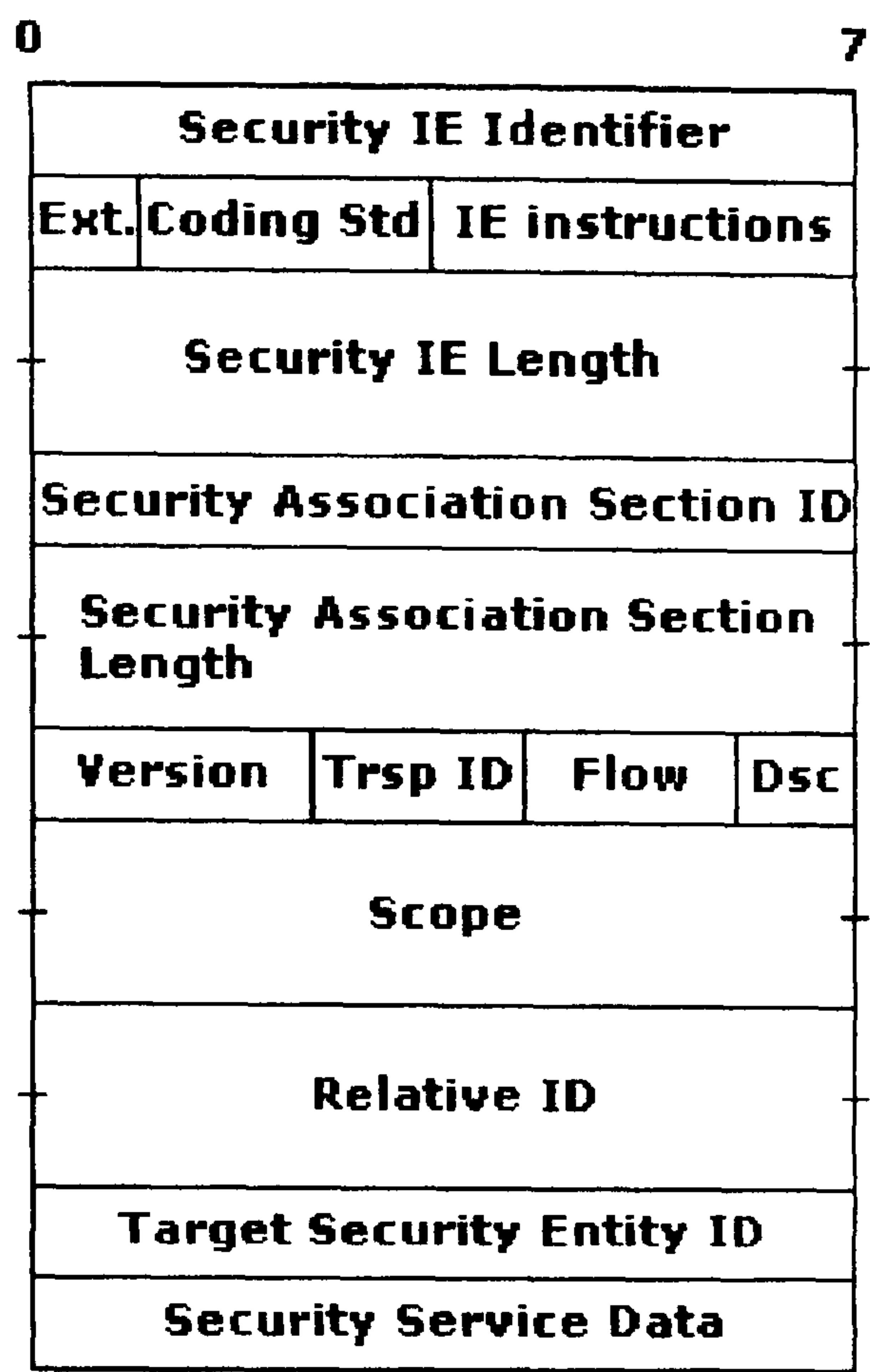


Figure 3.2: ATM Security Information Element

The security IEs are exchanged between ATM elements either in-band, that is, inside the ATM data stream, or together with the signalling messages of the NNI. The size of the IE varies from 12 bytes to 512 bytes, with security information taking up to a maximum of 508 bytes [96]. The *Security IE Identifier* field uniquely identifies the particular IE, whereas the *Ext.* and *Coding Std.* fields represent the specification used to realise the message (ATM Forum or ITU). The *IE instructions* hold information as to the action that should be taken upon reception of the IE, whilst *Security IE length* field corresponds to the overall length of the IE message. Security associations in ATMsec are identified via the *Security Association Section ID*, which is followed by an indication of the length of the section included in the *Security Association Section Length* parameter. The *Version* field denotes the version of the IE, with the *Trsp (transport) ID* depicting the type of transport used to exchange security information.

The *Flow* field indicates the flow number of the exchange whereas the *Dsc (discard)* block identifies whether the node should delete the association section before forwarding the security IE. Moreover, the *Scope* identifier includes information as to the node responsible for the processing of the association, with the *Relative ID* field depicting the security association ID. Finally, *Target Security Entity ID* is a parameter explicitly identifying the target node, with the *Security Service Data* section encompassing the type of security information sent.

The services offered by the architecture are authentication, integrity and replay detection. Authentication is realised by means of digital signatures included in the Security IEs data sections, supporting replay detection functions via the introduction of a timestamp in the algorithm calculation. Message integrity and confidentiality are maintained via encryption, either on an end-to-end or hop-by-hop basis between the security agents responsible for managing the security association. In either case, a dedicated channel (Virtual Channel – VC) should be set up to provide secured communication.

Placement of the security services is subject to the development of the respective specification by the standardisation bodies. Both the user and control planes should be afforded authentication and confidentiality functions to protect users against Denial of Service (DoS) attacks. The management plane is also of great importance, forming an appropriate location for maintaining the security services of the model. Since transmission is subject to the operation of the AAL and ATM layers, services should also be implemented in those parts of the architecture, to provide for confidentiality and integrity of the ATM data.

The disadvantages and difficulties in deploying the ATMsec, result from the complexity of the ATM networks architecture [97]. ATM switches are high-speed cell routers exhibiting fine grain multiplexing of the ATM messages towards the intended end-points. Due to this feature, it is very difficult to design and construct

high-speed cryptosystems that could be used to enable the real-time protection of the cells when traversing the VCs. To overcome this, network operators could provide a unique key for each VC protecting all information passed through it. The establishment of a security association on an end-point basis could further enhance the mechanism, with ATM cells being secured on the AAL level.

The challenge presented in this technique is the assignment of the protection mechanisms and their application in the data stream. On the ATM level, functions can only be applied on a single-cell basis, due to the mode of the ATM switch operation [97]. If confidentiality and integrity mechanisms were to be supported, then every cell must be encrypted and decrypted, increasing the processing time overhead in the network. Furthermore, current cryptosystems can barely address the above requirements on speeds of the order of hundreds Mbps or Gbps. Additionally, the secret keys used for the VC encryption should change rapidly but no mechanism currently exists that could update the keys as sufficiently and fast.

The proprietary nature of the ATM networks, as well as of the hardware cryptosystems available, could increase interoperability and compatibility problems between different environments. The respective ETSI/3GPP specification on the use of ATM in the CN [98] does not include any security considerations for this transport mechanism. The high-speed requirements on software and hardware based solutions prevent the use of ATMsec in realistic network topologies.

3.3.1.2 Stream Control Transmission Protocol (SCTP) Security

The Stream Control Transmission Protocol (SCTP) described in Chapter 2 has been designed to provide a transport alternative to TCP for signalling data. Its use is recommended in cases when small amounts of information (of the order of a few hundred kilobytes) have to be transferred between two end-systems, inside networks such as the UMTS CN.

SCTP has been developed to overcome security pitfalls of the TCP associated with exploitations of the handshake mechanism and replay attacks. Although the provision of an enhanced connection establishment mechanism ensures the connection integrity, there have been several deficiencies identified, as presented in [P1] of Appendix C.

SCTP does not implicitly compensate for security functions, like the IP protocol architecture. Instead, it has to rely on the use of a transport layer mechanism to address authentication and confidentiality issues in the SCTP associations [99]. Although the handshake mechanism of the protocol has been designed to overcome problems associated with masquerade and flooding attacks, the lack of security mechanisms implies that the messages could be sent in cleartext format. If that is the case, then an unauthorised party monitoring the communications channel could launch man-in-the-middle attacks, whereby information would be intercepted resulting in data fabrications. The lack of appropriate functions affording confidentiality and integrity measures could jeopardise the SCTP association and message continuity.

If the connection is not supported by the IPsec architecture, then problems associated with forged messages could be dramatically increased. An attacker could mount a masquerade attack at the network level, denying the transmission of the SCTP messages, ultimately causing a DoS attack. On the other hand, the provision of the IPsec architecture would imply the use of several Security Associations (SAs) because an SA is only a simplex connection. Should multihoming be supported then the number of SAs would rapidly become unmanageably large as the network grows.

The provision of connection establishment functions overcoming vulnerabilities of the TCP protocol is subject to the use of mechanisms explicit to the SCTP. The lack of proper security definitions in the SCTP specification could result in an increased

complexity when deploying a system, due to the need for additional techniques protecting the association. The impact on the flexibility and compatibility of the mechanism could result in the choice of alternative sets of communication rules for transporting signalling information in mobile environments.

Finally, the lack of a functional SCTP stack for use within UMTS CN and computer network systems renders the protocol use vague. The possibility of additional security requirements in mobile environments [100] could result in modifications of the SCTP specification, leading to interoperability problems and more complex architectures. Failure in conforming to standards supporting security functions and services for mobile systems [101], [102] make SCTP an unattractive solution for use within the UMTS CN domains.

3.3.1.3 IP Security Architecture (IPsec)

IPsec defines a set of security mechanisms and procedures that can be used to secure communications between two hosts at the network level [103]. It is a set of communication rules designed to act independently of higher layer protocol functions compensating for link-based protection services for the data stream.

The provision of security functions on the packet level has made IPsec the foundation for the establishment of Virtual Private Networks (VPNs) [104], [105]. Companies and other private organisations are able to set up secured dedicated communications channels, defining the appropriate policies governing their connection as to the traffic allowed to traverse the network.

IPsec operation is based on the Authentication Header (AH) [106], Encapsulating Security Payload (ESP) header [107] and the Internet Security Association and Key Management Protocol (ISAKMP) [108]. The AH and the ESP form part of the extension headers of the IPv6 protocol and their implementation is subject to the

requirements set by the network operator. The AH is used to provide data integrity and data origin authentication, together with a degree of protection against replay attacks. Confidentiality is a security service not defined for AH, and hence no encryption takes place on the IP packet. Figure 3.3 illustrates the format of the AH header, which should directly proceed the IPv6 header if no other options are applied (Table 2.3, Chapter 2). Additionally, ESP usage ensures the confidentiality of the information exchanged, together with data origin authentication and data integrity. Figure 3.4 depicts the general format of the ESP header.

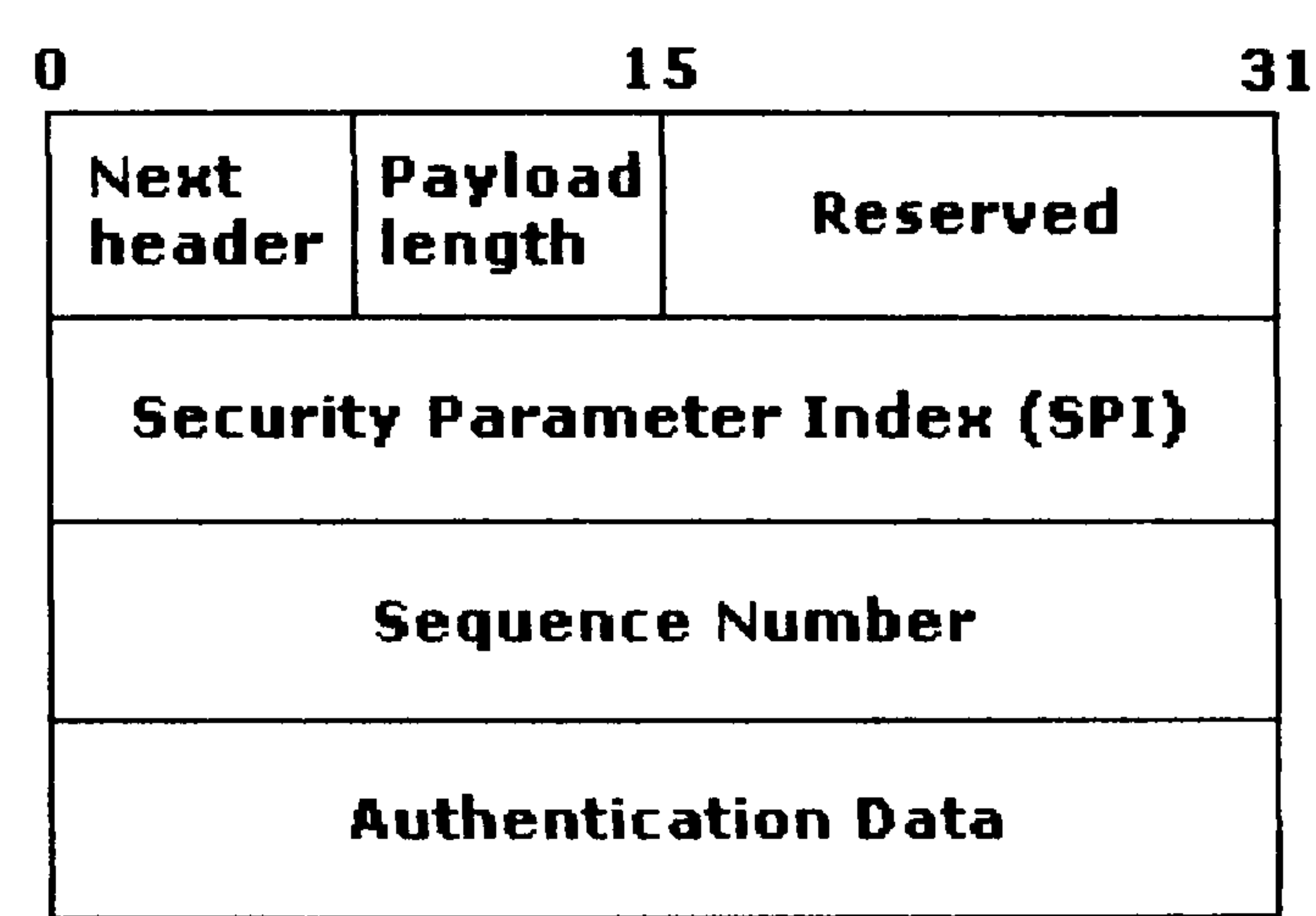


Figure 3.3: AH header format

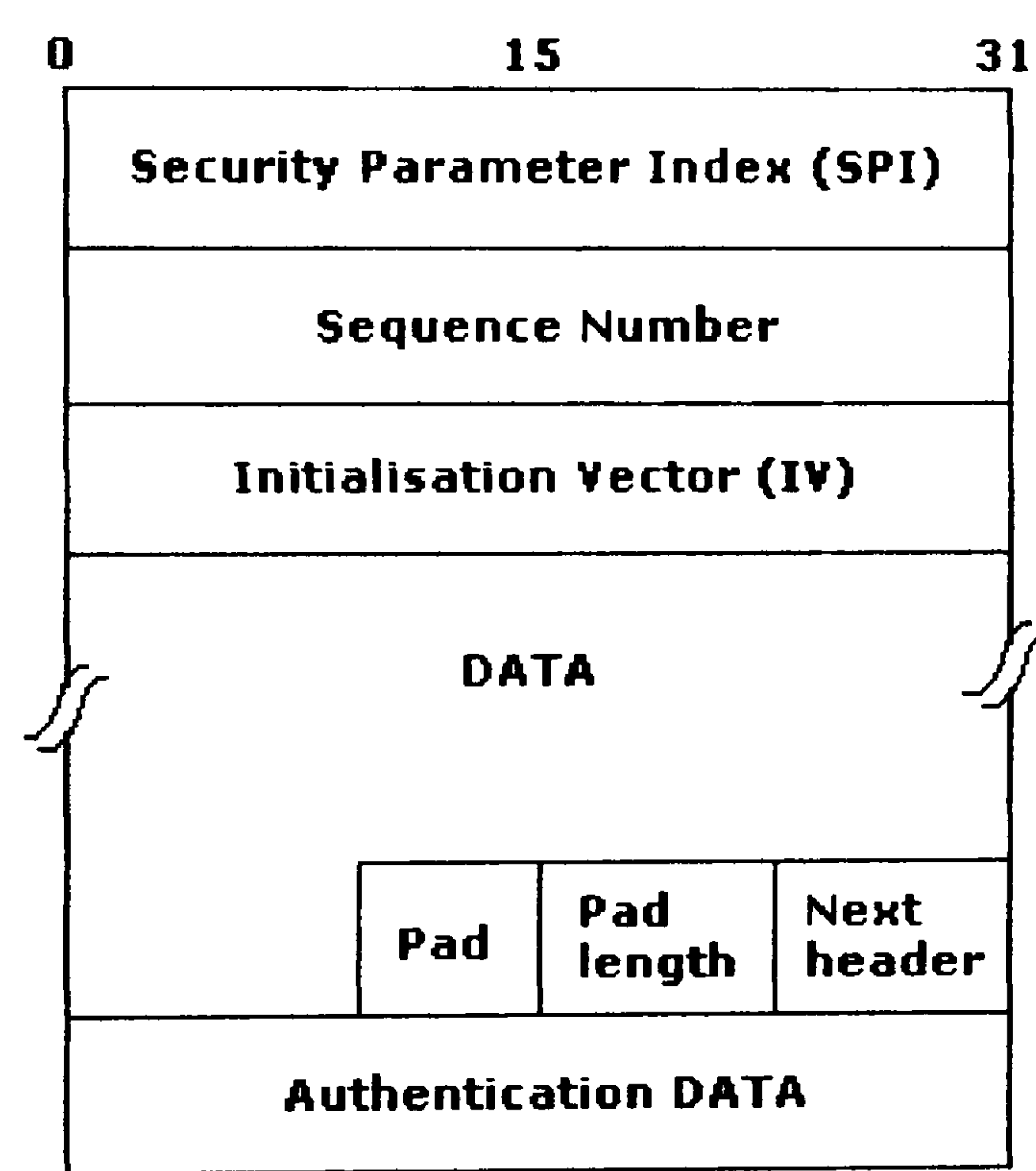


Figure 3.4: ESP header format

The *Security Parameter Index (SPI)* field is used to indicate the SA identification to the receiver, depending on the security header implemented. The destination should use this value to access the *Security Association Database (SAD)* and distinguish between the different SAs set. To verify that the required privileges are preserved in the received packet, the receiver associates this entry with the contents of the *Security Policy Database (SPD)* to decide upon the acceptance or rejection of the message. For the AH header, the *next header* field represents the higher-level protocol following the AH, such as the TCP. To enable error detection procedures, the *Payload length* block holds the length of the AH header, whilst *Reserved* bits are unused and set to zero. The *Sequence number* is a measure for

supporting packet sequencing and for providing a countermeasure to possible replay attacks. Finally, for the ESP header, the *Initialisation Vector (IV)* field represents the value driving the cipher algorithm used to apply integrity mechanisms for the transmitted packet.

The specification of IPsec defines two modes of operation for the architecture. For both cases, an SA should be set between nodes to manage the exchange of the secured information.

In *Transport Mode* the AH and ESP are used to protect the data stream in cases where the SA end-points are the actual communication end peers. No intermediate nodes should be present in this mode, with the security headers protecting the transport payload data from a higher layer such as the TCP. The AH provides authentication for the whole message, whereas the ESP supports both confidentiality for the higher layer header and data authentication for the message apart from the IPv6 header.

In *Tunnel Mode*, IPsec services are afforded to the user data whenever the communicating end-parties are not the SA end-points. This implies the presence of intermediate routers in the topology, increasing the number of SAs that need to be set. With the ESP security enabled the entire IP packet, including the higher layer data, is protected. The ESP adds an additional IP header to enable routers identify the next hop message should be forwarded to. Similar to the ESP operation, the AH in tunnel mode authenticates the whole IP packet adding an external IP header to assist routers in appropriately switching the received messages.

If both the AH and ESP protection are applied, then two or more SAs must be created to support the communication. If such an SA is not available, then the ISAKMP protocol should be used to create one via the Internet Key Exchange (IKE) mechanism [109]. IKE defines the key exchange process and the parameters of

the security services required. The process is initiated by defining an SA that is used to negotiate the SAs for the IPsec. The peers then use the Domain Of Interpretation document to specify the exact contents and algorithms of those parameters. Once the necessary parameters have been set, communication can proceed having afforded security measures on the network level [110].

IPsec advantages encompass the addressing of security vulnerabilities present in the IPv4 architecture. The provision of protection mechanisms in the routers of a network topology has signalled the long awaited security consideration of the QoS levels for the subnetworks supporting the transport end-to-end communication service. If a network operator chooses to apply IPsec for a connection, then any traffic not complying with the SPD set will be discarded. This method ensures that unauthorised data streams will not bypass any IPsec firewall, protecting the nodes present in the communication.

Additionally, IPsec functions are visible only to the underlying subnetwork since they are placed below the transport layer. This transparency towards the applications increases the independence of the technique from the users involved in the exchange. The processing time that would consequently be required to identify legitimate packets by the node instances is decreased, since it falls to the routers to recognise and permit lawful packets. Moreover, it provides security for individual users by specifying explicit SAs for particular applications and attacks related to fabrication and data modification are countered.

The advantages of IPsec leveraged the design of the UMTS CN NDS for the Release 4 and above addressing privacy issues in mobile environments [111]. Independent hardware vendors have investigated IPsec in solutions regarding security for the CN [112], [113] and in network routers for general PS environments [114].

However, the IPsec operation has several disadvantages that may influence its implementation. At the time of writing this thesis, most of the IPv6 stack where IPsec is based has not been completed [115]. The lack of a functional stack prevents vendors in developing test beds for the IPsec that could be used to measure the protocol performance in real-network situations. Organisations supporting the deployment of IPv6, such as the European Commission IPv6 Task Force, have addressed the need of migrating from the currently used IPv4, for which though IPsec is not a standard. Although operating system architectures such as Microsoft Windows 2000/XP and Linux families provide an initial implementation, network operators have yet to offer security services on the network level.

Specification pitfalls influencing the functionality of the IPsec are the complexity and lack of clear conditions of the IPsec documentations [116]. Complex systems are deemed to exhibit more complex failures, which are hard to fix and manage. The weaknesses that may be present in the design are difficult to identify and analyse leading to potential vulnerabilities. The presence of many options in supporting the realisation of the IPsec may create confusion as to the mechanisms network operators could use. This is due to the fact that the system designer has no way of determining the functionality of the architecture, also partially due to the lack of a clear service definition.

Ferguson and Schneier in [116] have also identified several deficiencies related to the operation of the AH and ESP headers of the IPsec, as well as of the ISAKMP and IKE. It has been suggested to remove the AH from the specification because it does not provide for packet encryption but only authentication. As a proceeding step, [116] addresses the possibility of eliminating the IPsec transport mode of operation, which forms a subset of the tunnel mode. The functionality of the ESP architecture can easily match, in terms of bandwidth (BW) usage, that of the AH and hence become the only protocol supporting the security services. At the same

time, the removal of the transport mode would eradicate the need for categorising the network nodes in hosts and routers (security gateways in IPsec terminology).

Additional measures could apply for the ESP modification to support both authentication and message confidentiality, since encryption without authentication is not useful. The network nodes should authenticate the plaintext instead of the process currently defined, where authentication takes place on the encrypted portion of the packet. Finally, since encryption/decryption are techniques based on the use of a secret key, the system should seek authentication in these blocks and in every parameter that would be used to interpret the packet.

Disadvantages have also been identified for the handling of the Internet Control Message Protocol (ICMP), which is described in Chapter 6. A note is made on the discarding of unauthenticated ICMP messages in [112]. If a router is unaware of the SA used, then all ICMP messages sent will be in plaintext format. Dropping such messages would consequently have an impact on the IP stack functionality that may even cause the termination of the communication. Should the router use IPsec then an additional SA will have to be established. The problem associated with this approach is that the tunnel endpoint will seek verification at the message source. Due to the lack of a functional Public Key Infrastructure (PKI) and the disadvantages identified for its use [117], the SA setup will not be possible.

Several other implementation pitfalls have been acknowledged for the ESP security algorithms and the ISAKMP [118], mainly resulting from the inadequacy of the respective specifications. Exact details of the drawbacks of the encryption algorithms on a mathematical level fall outside of the scope of this thesis but it is worth noting the research presented in [119 - 125]. Most of the attacks presented in these papers relate to the use of encryption mechanisms for the IPsec, as well as disadvantages acknowledged by the use of mobile IP extensions. Again, the vast majority of the vulnerabilities presented result from the lack of clear definitions in

the IPsec specifications, whereby too many options can be used for the implementation of a single service. Some of these attacks have been presented in [P2] of Appendix C.

The major disadvantage of the IPsec recognised in this work has been the extensive use of additional mechanisms to secure the communication process, which are usually explicit to the communication protocols. Although vulnerabilities associated with IPv4 and the TCP/IP suite, such as the IP address spoofing [126], have been overcome, the protection of the user data still relies on external architectures. Problems associated with the stack architectures supporting PS networks [127 - 130] have led to the development of supplementary mechanisms for the support of the IPsec infrastructure [131], [132], increasing the complexity of the resulting system and possible interoperability problems between network operators.

The identification and placement of security mechanisms in protocol stacks are issues not addressed by the IPsec, which forms a layered protocol placed directly below the network layer of the TCP/IP suite. Research has been focused in increasing the performance of current encryption algorithms [133 - 138] but without providing an adequate framework for their use. The only attempt made to identify security functions in protocol architectures has been the OSI security model, described in the following section. It should be noted however that OSI defines a reference architecture and not a functional stack.

3.3.2 OSI Security Model

The architecture presented in [84] represented attempts made to place security functions inside the communication layer protocols of the OSI model [139]. Further research undergone concluded in the inclusion of mechanisms such as authentication and non-repudiation in specific layers of the protocol stack [140], to enable the ease in their management and the avoidance of functionality duplication.

This can be observed in Figure 3.5, which depicts the placement of the security mechanisms within the layered protocol environment.

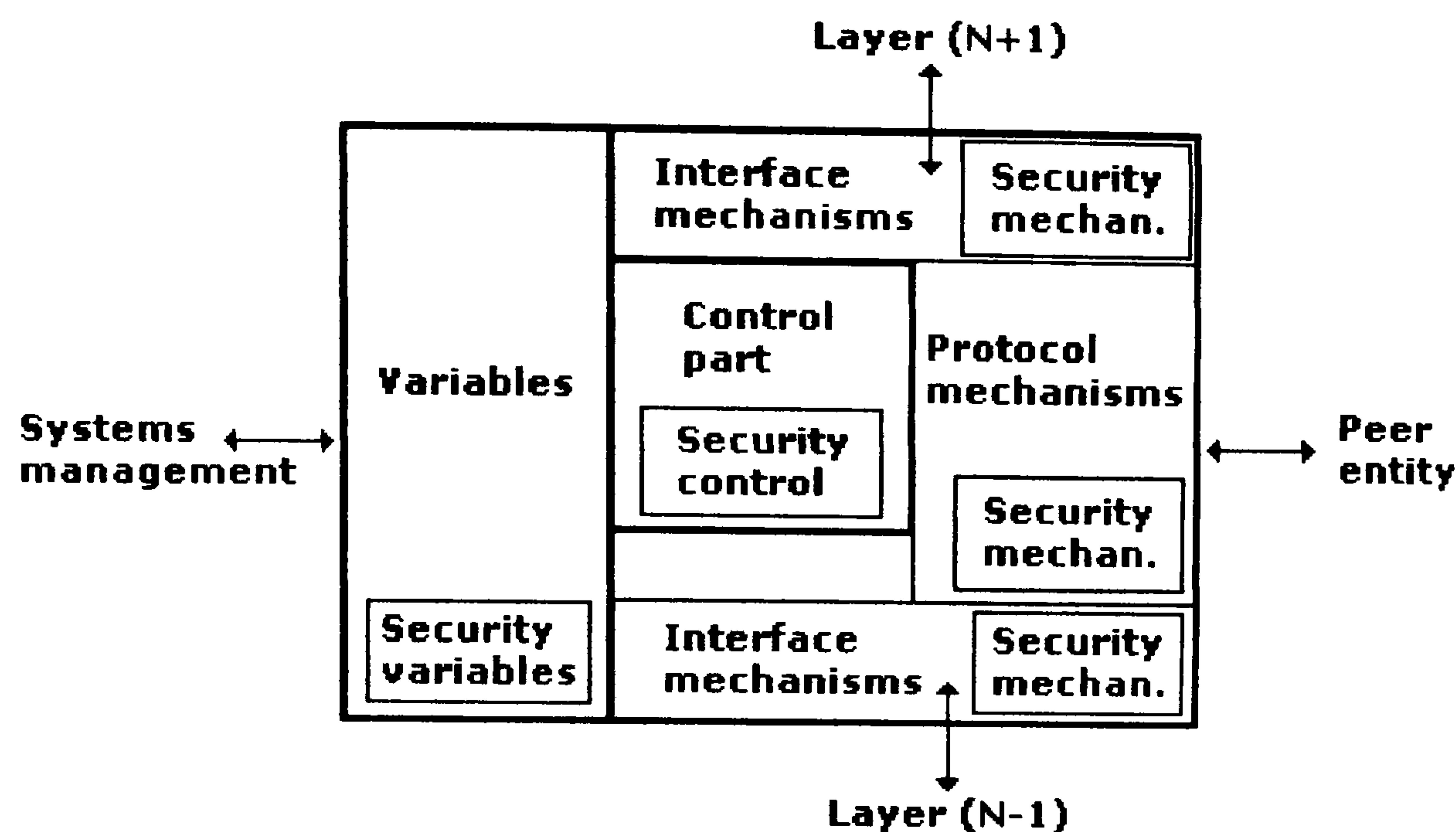


Figure 3.5: Layer N security architecture

The approach is based on inserting any necessary mechanism inside the specific communication layer entity. The protocol will consequently have the tasks of both managing the functions responsible for the connection maintenance phases, as well as those for the realisation of the requested security services. The interfaces and variables are held within the respective areas of the protocol layer and are exchanged between the peers and/or adjacent layers accordingly [141].

The assignment of security functions into the OSI layers has followed two directions. Figure 3.6 and Figure 3.7 indicate those as taken by [84] and [140]. The structure presented in Figure 3.6 identifies the importance of the security functions within the OSI model, whilst Figure 3.7 shows the position of these in the stack architecture. Their categorisation has been based on the method data is forwarded down the stack towards the intended destination and the layers that could have an impact on the security of that data. As an example, authentication has been placed into the application layer, since it is strongly related to the nature of the information exchanged and the end-users initiating the connection.

OSI 7 layer model

Application layer
Presentation layer
Session layer
Transport layer
Network layer
Data Link layer
Physical layer

OSI security model

Authentication
Access Control
Non-repudiation
Data Integrity
Confidentiality
Assurance/ Availability
Notarisation/ Signature

OSI security functionality

Authentication Non-repudiation Access Control	Application
	Presentation
	Session
Data Integrity Confidentiality Access Control	Transport
Access Control	Network
Access Control	Data Link
	Physical

Figure 3.6: OSI security model mechanisms**Figure 3.7: OSI security functionality**

In this thesis several deficiencies related to the OSI security framework have been identified that led to the design and development of the Future Core Networks System (FCNS). Current research papers on OSI security such as [85] focus on attacks based on programs and applications representing the layers of the reference model. Characteristic results include the physical layer exploitations based on disrupting power sources and cutting down cables, or exploitations such as Unicode vulnerabilities (for the presentation layer) and viruses (application layer specific). The provision of mechanisms ensuring the protection and recovery of the system against such attacks is explicit to the operation of the layer protocols and the internal to the stack operation messages. An attack could be launched against the mechanisms a layer uses to communicate with the protocols adjacent to it, with devastating results for the connection.

The mechanism of Figure 3.5 has been designed to enable the maintenance and possible update of a protocol's functionality, without affecting that of the remainder of the stack. Implicit functions increase the flexibility and portability of the architecture, simplifying also the design of the stack by defining strict rules for its modularisation.

However, this approach fails for two major reasons. As is depicted in Figure 3.7, access control functions could be placed on several layers of the OSI model, since services might be offered at numerous levels of the communication process. Depending on the policies governing a system, network operators could choose to limit access privileges on network routers by filtering out unwanted addresses (network layer), or at the transport layer by blocking illicit ports. The application of access control functions for several layers would result in the duplication of the functionality for the different protocols supporting the service. Although the principles determining the access rules would vary, the mechanisms used to support the offered service would be similar if not the same. If the system was ever to be updated, the designer would have to repeat the same operation for the number of protocols to which access control functions would be afforded.

Secondly, current advances in communication protocols have addressed the need for service realisation in single layered structures, such as the IPsec. The AH and ESP headers provide authentication, message integrity and confidentiality, which disagrees with the OSI reference model principles. Again, the policies supporting the security functions of IPsec could be different than those used for the application. Still though, functionality duplication would not be avoided, if for example both the application and the subnetwork operator choose to authenticate their messages. The amount of processing time and effort required to update and modify the stack architecture would decrease the system's portability and flexibility in adapting to different environments.

Additionally, the placement of security mechanisms inside the communications layers increases the processing load of the layer protocol instance. As can be seen from Figure 3.5, the amount of calls a layer should manage is doubled due to the protection functions residing in its core. Additional PDUs and interface access points are required to support the protocol operation. Further to this demand, the protocol entity may call for additional CPU load to support the system calls and

processes set by its users (the layers above it) due to the addition of further parameters.

Furthermore, the OSI security architecture fails to address an efficient Security Context (SC) exchange mechanism. Suppose that the user is establishing a network connection between peers belonging to different subnetworks. If the necessary mechanisms fail to exchange the security parameters to be used for the data exchange, then that would imply the immediate termination of the process. Although this is a required action, security mechanisms failure could mean a failure in the layer mechanisms as a whole. The network operator will not have an immediate view of the error severity and the exact location in which the fault occurred. If that is the case, then the layer entity should be reset further delaying the connection establishment process. Even if the protocol was to recover without losing functionality, it would need additional time in clearing the security variables and parameters arranged.

Finally, the architecture fails to direct the requirement of management simplicity and robustness. Security functions placed on numerous layers of the stack architecture increase the complexity of the system. Potential errors and design problems are more difficult to detect and correct, complicating further the system's maintenance. As far as security is concerned, complex systems do not necessarily entail more secure ones. On the contrary, complex structures may contain deficiencies and vulnerabilities that may be identified only by their exploitation by an unauthorised party once they have been realised in a real-network environment.

Chapter 3 has summarised the main concerns forming the motivation behind the work presented in this thesis. The FCNS has been designed to overcome the problems associated with the out-of-date specification of the OSI reference model in providing for the security of the communication process. The Security Layer (SL) application on the whole of the FCNS communication layers, results in simplified

network management procedures and ensures that a failure in initialising the security parameters requested will not affect the operation of the stack architecture. Figure 3.8 represents the security architecture of the layered entities of FCNS.

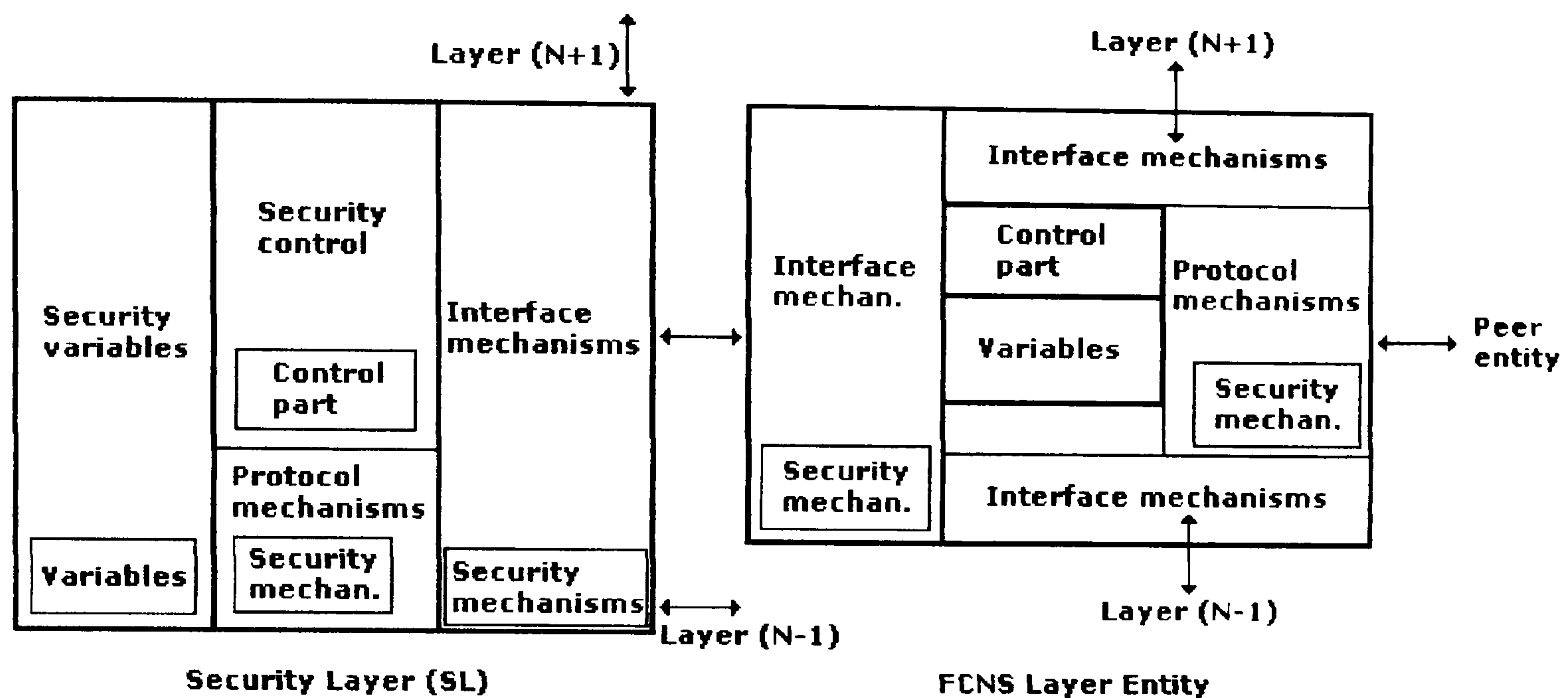


Figure 3.8: FCNS Layer N Security Architecture

The FCNS design removes the implicit security functionality of the OSI layers, transferring it into a single managed protocol layer. Communication layers are only involved with the conformance to the algorithms and functions enforced by the SL, included in the respective SC. Variables and parameters, as well as security control functions reside only in the SL structure, thus removing any redundant information from the rest of the layer entities.

The benefit of this approach is the preservation of the layered principles, where functions specific to a layer are included only in that protocol. Since the OSI model dictates that any functionality duplication should be avoided, this work has removed any security related mechanisms from the layered protocols involved in the communication process. In case amendments need to be made due to possible advances in cryptosystem technology, the designer should only focus on the SL, leaving the rest of the stack intact. By this method, the processing time and effort it would take to update the stack entity is reduced compared to that required for an

architecture referencing the OSI security model. Similarly, changes in the communication layers will not affect the operation and functionality of the SL.

Furthermore, the architecture enables the operator in quickly identifying the location of any errors that might occur. The clear distinction of the security and communication functionality of the FCNS structure enables the system to unambiguously locate error conditions. Faults specific to the SL would not imply an erroneous condition at any one of the FCNS communication layers and vice versa. By this method, the isolation of the faulty part can be made possible, so that the continuation of the communication process is preserved.

Finally, the FCNS security architecture increases the flexibility and portability of the system, as well as the degree of adaptation in various network environments. The preservation of the communication layers functionality enables the network operators to use any currently available protocols to serve as one of the FCNS layers. The performance of the system is therefore enhanced due to the method used in providing the security functions to the communicating entities. The simplicity of the FCNS design results in the ease of maintaining the rules governing the operation of the stack and the identification of any possible implementation pitfalls from the design process. At the same time, security functions realisation is a recommended action, which should only be used as an additional measure and not as a prerequisite for the operation of the network. This implies that if the SL functions are ever needed, then the protocol will provide those to the entity requesting the appropriate mechanisms. On the other hand, if such a provision has already been made (for example the IPsec architecture), then the SL can either be omitted from the implementation or reside in an idle state waiting for inclusion.

Chapter 4: Future Core Networks System (FCNS) - Communication Layers

Chapter 4. Future Core Networks System (FCNS) – Communication Layers

4.1 Overview

4.2 FCNS Architectural View

4.3 FCNS Functional View

4.3.1 User-defined Presentation Layer

4.3.2 User-defined Session Layer

4.3.3 Transmission Layer

4.3.4 End-to-End Layer

4.3.5 Physical Layer

In this chapter, a description of the FCNS architecture and its layers with respect to the OSI 7-layer model [139] is provided. The FCNS layers are analysed and explained in detail, in connection with the FCNS protocol and service specification and their role in its implementation. The layers' protocol definition subsections include information as to the functions each layer performs, whilst the service definition subsections indicate how the services offered by the protocol functions are realised. The differentiation between functions and services has been made to clarify the operation for the reader, since their notions are strongly dependent, in the sense that the services' realisation via the protocol functions defines the complete specification of that protocol. Management and analysis of the FCNS security functions are given in the Security Layer (SL) protocol specification in Chapter 5, since this is responsible for their enforcement and monitoring.

4.1 Overview

FCNS has been developed to provide secure user and signalling data exchange between network elements belonging to the same network or across various subnetworks in a packet-switched architecture, such as the UMTS core network Release 99 [3]. Telephony and wireless voice access systems are circuit-switched oriented and hence FCNS applicability in such systems has been outside of the thesis scope.

FCNS offers a reliable end-to-end acknowledged connection – oriented service, with congestion and flow control mechanisms regulating the information flow in the network. Connectionless services have been included in the FCNS design mainly in its security part, incorporating functions such as data origin authentication and connectionless integrity. Moreover, FCNS incorporates error detection and correction mechanisms together provided by the FCNS Error Protocol (FCNSEP), affording a secured mechanism for identifying transmission errors and fault reporting.

Finally, FCNS features its own keystream generator to provide the secret keys necessary to secure the service primitives of the protocol and all inter-layer messages exchanged in both sender and receiver. The generator has been designed to be independent of the network on which FCNS is used, enhancing the protocol's flexibility and scalability.

The FCNS keystream generator is described in Chapter 5 and FCNSEP in Chapter 6, with the design principles of the protocol being given in Chapter 7 together with information concerning its prototype and source code implementation.

4.2 FCNS Architectural View

The FCNS stack is a layered architecture designed according to the OSI model principles, to ensure compatibility with current architectures. Its conceptual view is depicted in Figure 4.1, while Figure 4.2 illustrates its comparison with the OSI architecture.

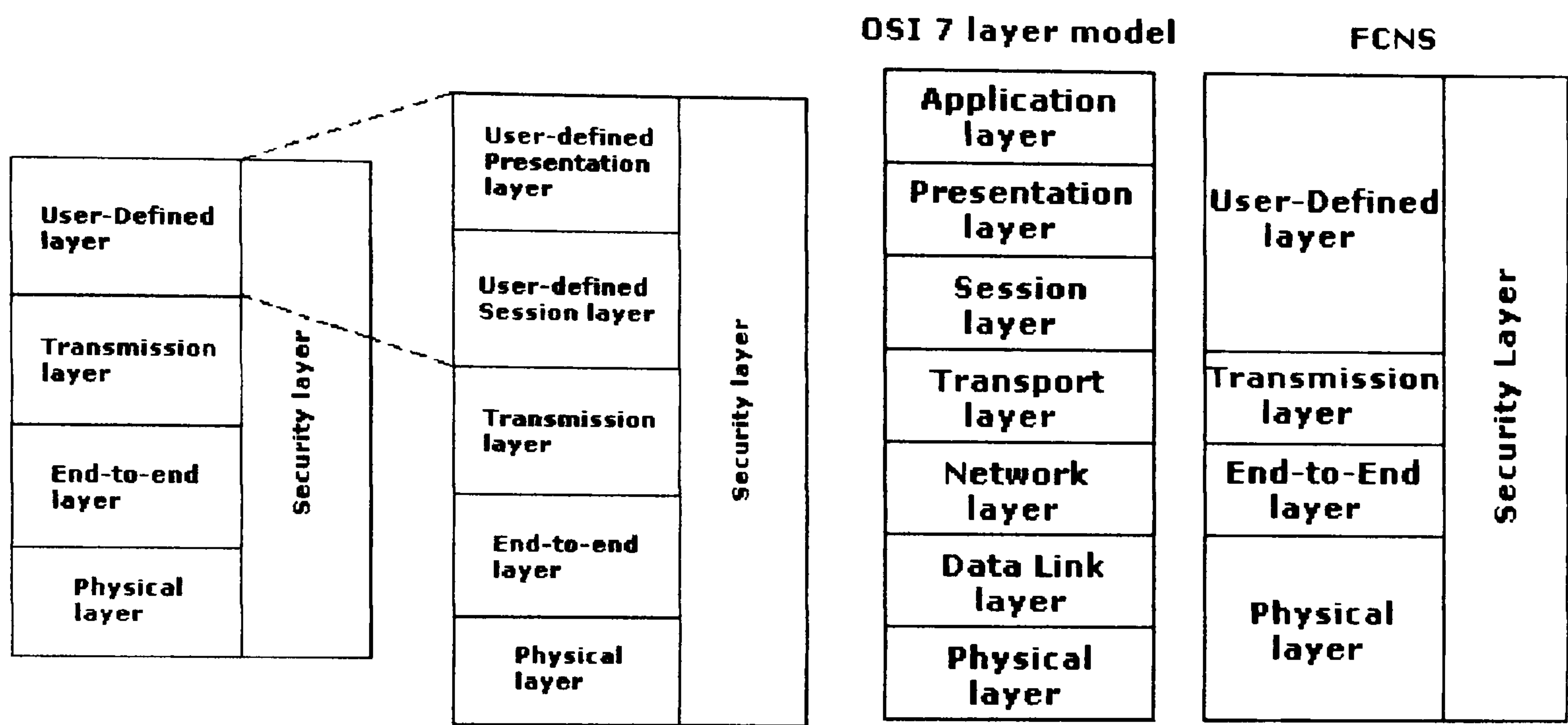


Figure 4.1: FCNS conceptual view

Figure 4.2: FCNS and OSI model

FCNS does not have a definition of a physical layer similar to that of the OSI model. The reason for this approach has been that the simulation of that particular set of communication rules would not have provided any valuable information as to the security of the service primitives and data exchanged using FCNS. Although the lack of an adequately secure physical medium forms the motivation behind the development of the cryptography research area, the design of such a layer has fallen outside the scope of this Ph.D. Thesis. Assumptions were made about the environment on which FCNS runs, such as the use of optical fibres that cannot be easily tapped and other well known standards (such as RS232 and V.35) for raw data transmission.

FCNS has been designed to provide services and functions to any application, irrespective of its nature. There is thus also no inclusion of an application layer as defined by the OSI model, since that would imply strict definitions that are outside

the scope of this work. However, for the simulation needs of this thesis, the layer's functionality has been simulated, to identify and test security functions such as authentication and non-repudiation, which are usually application dependent and hence vary between several different communication sessions. Their management though has been left with the FCNS security layer, whose design is sovereign of the rest of the FCNS layers or any other protocol that can be used together with the stack, something that had a catalytic effect in the omission of the application layer from the design. Figure A.2 of Appendix A illustrates the specification diagram created for the needs of simulating the sender and receiver peer instances running on top of FCNS.

The most important observation that is made by the FCNS conceptual view of Figure 4.1 is the position of the *Security Layer (SL)* with respect to the rest of the stack structure. The main implication of this design is that SL is used to intercept any messages that are to be transferred, either between the FCNS layers or peers and at the same time manage the security functions available to the FCNS implementation. This does not have any effect on the way communication should proceed, since although the use of the SL functions is recommended, it is not compulsory. There are cases where security may not be needed for a specific layer, such as the data link layer in cases of real-time video transmission, and consequently a network operator may choose to disable the feature for that particular session.

Communication using FCNS follows the principles identified by the OSI model, as illustrated in Figure 4.3. The schematic represents the interaction between two peers, belonging to different subnetworks. FCNS takes care of the routing and security procedures required, regardless of the number of intermediate nodes, to ensure a protected and in-sequence message transmission from the initiator to the destination node.

The SL, when fully applied, in addition to enforcing the necessary security functions, has the task of identifying whether the messages have been successfully encrypted prior to their sending, as well as when intercepting them at the adjacent routers. For a particular connection, the appropriate algorithms and secret keys are initiated, exchanged and maintained by the Security Context (SC) exchange process managed by the SL.

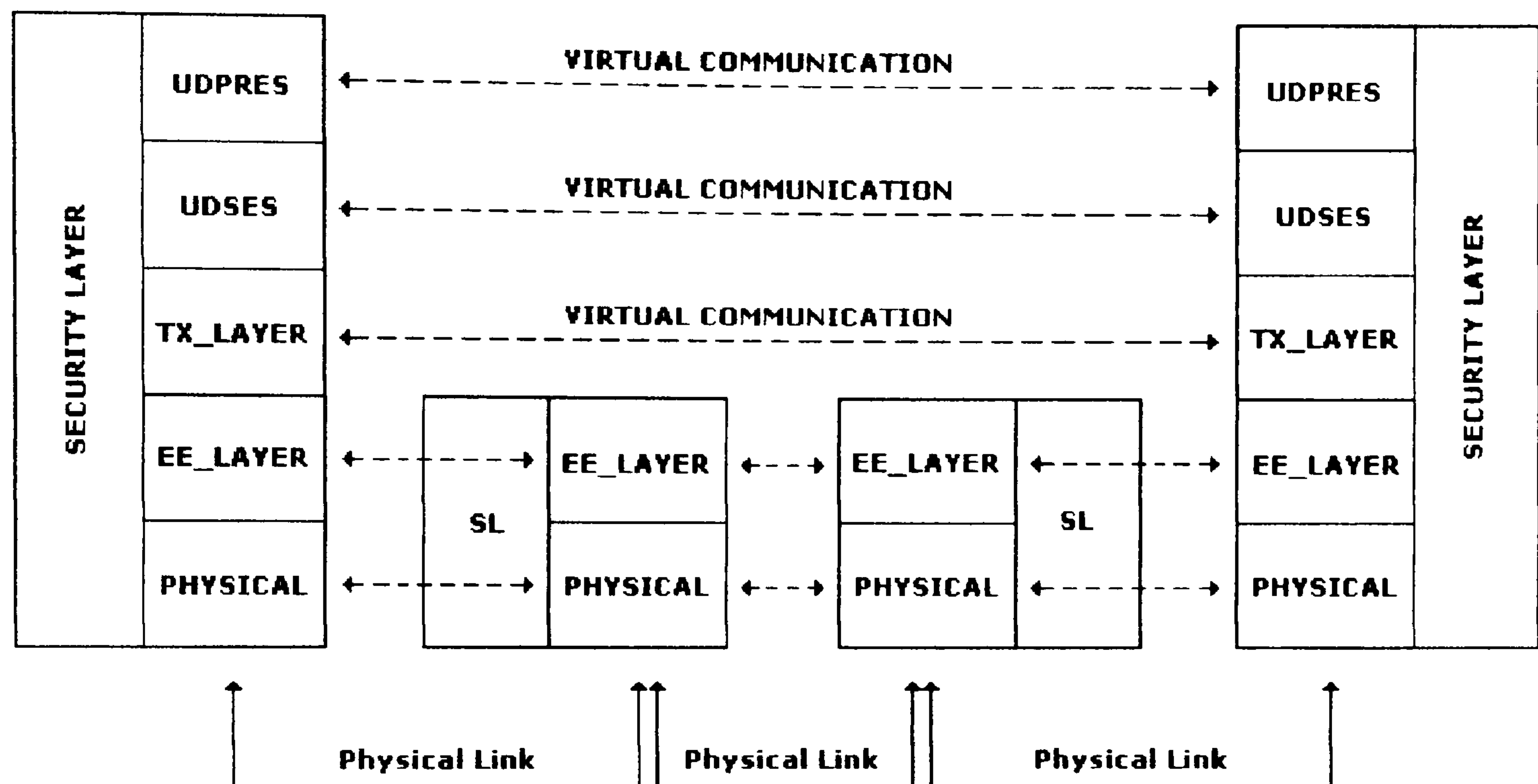


Figure 4.3: Two peers communicating using the FCNS

For simulation and development needs an adequate and efficient key exchange and management protocol has been assumed when necessary, since it is likely that key exchange in the networks in which FCNS would apply would be managed by higher-level applications mapped onto the FCNS messages. For example, in UMTS, encryption for the external messages to the FCNS operation would be secured using the UMTS encryption algorithms, with internal FCNS messages relying on the FCNS keystream generator.

The communication between the FCNS layers has been analysed and materialised by having each one of the FCNS protocols act as an individual set of communication rules, as illustrated in Figure 4.4. In this view, the links between the layers are actually virtual-based, in the sense that no errors can be inserted to them, since

these can only be initiated due to a software or hardware fault, a case irrelevant to the design and realisation of FCNS. The delays introduced in the FCNS operation are produced solely by the protocol stack functions, such as the message encryption/decryption processes, message encapsulation, fragmentation, reassembly and the like.

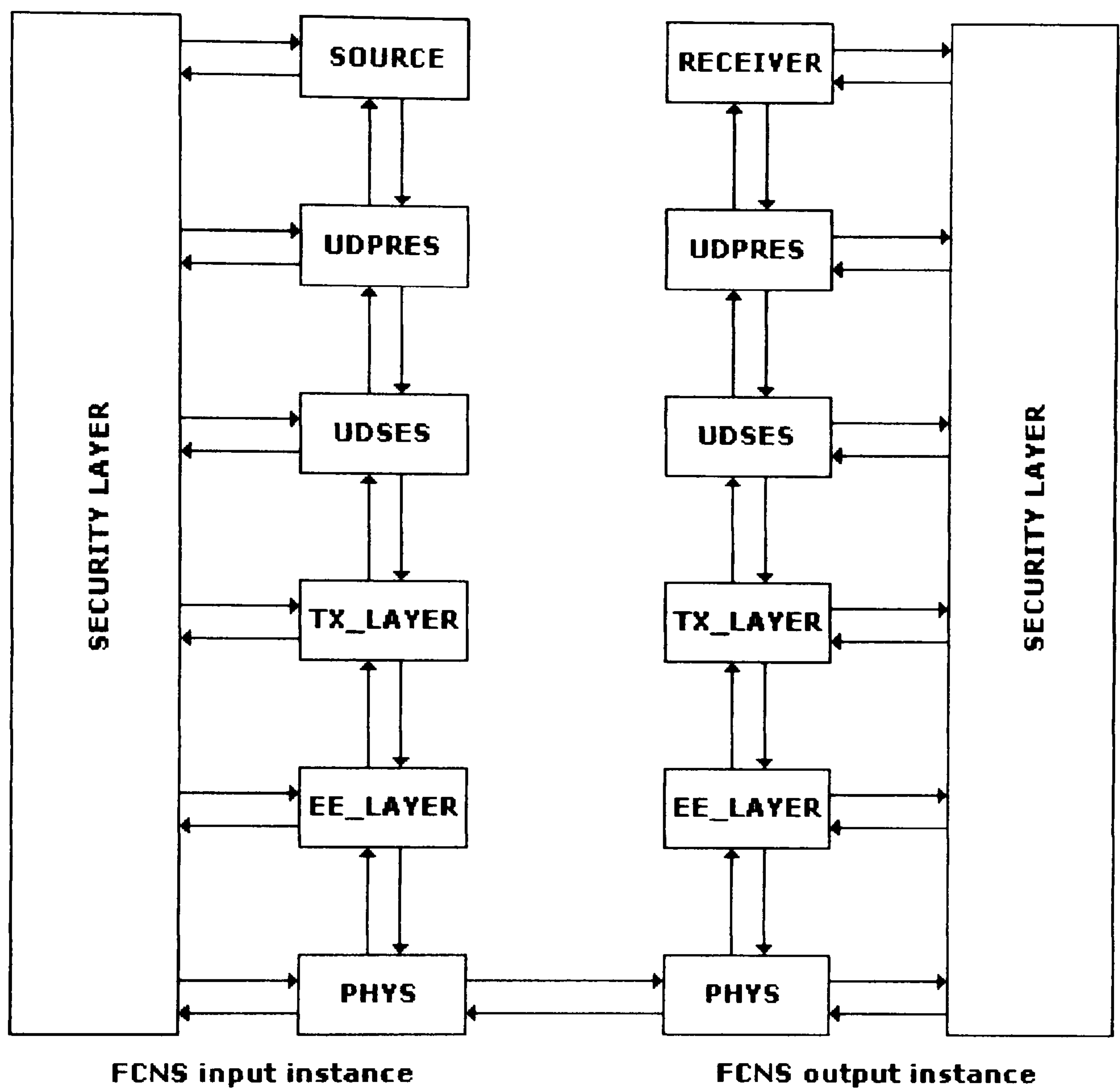


Figure 4.4: FCNS internal communication structure

The *SOURCE* and *RECEIVER* blocks represent the instances of a generator and sink that have been designed to provide for the necessary input and output node instances of a system, serving much like an application layer on top of FCNS. Their use is the initiation and tearing down of a connection or service, via the service primitives that will be exchanged between the FCNS instances. In a real-network situation, both the sender and receiver (and any intermediate subnetworks) will run both input and output instances of the protocol stack, since peers shall be able to send and receive data concurrently.

The User-Defined Layer of the FCNS is subdivided into the *User-Defined Presentation Layer (UDPRES)* and the *User-Defined Session Layer (UDSES)*. As with the respective layers of the OSI model, UDPRES has the task of ensuring the correct encoding of the information that is to be exchanged via the presentation context management services. The UDSES layer has the undertaking of synchronising a particular session, whilst handling the data octets transparently, that is, irrespective of their syntax.

The *Transmission Layer (TX_LAYER)* is involved with the provision of an end-to-end reliable service, given the Quality of Service (QoS) demanded by UDSES. This layer also implements an explicit flow control mechanism and provides error detection, error recovery and multiplexing capabilities. Segmentation, reassembly, concatenation and separation, as well as stream resynchronisation takes place in TX_LAYER, which has been designed as a connection – oriented service provider, although connectionless services can also be supported.

The *End-to-End layer (EE_LAYER)* has the task of affording the necessary routing mechanisms for internetworking purposes. It is also responsible for the reliable transfer of the data messages between two directly linked nodes, providing at the same time error detection and correction mechanisms.

The *Physical Layer (PHYS)* has the responsibility of transforming the datagrams passed by the EE_LAYER into frames, incorporating control information used to regulate and synchronise the message flow. It is the layer where implicit error and flow control mechanisms are implemented, such as the Automatic Repeat Request (ARQ) protocols.

The SL, apart from intercepting the messages exchanged in the FCNS architecture ensuring their protection, is the layer that integrates the FCNS Error Protocol (FCNSEP) initiating at the same time the FCNS keystream generator that provides

the secret keys used to secure the protocol's internal messages. The SL has to make sure that the same secret keys are not used by different sessions, while guaranteeing that link and end-to-end encryption keys are independent of each other. It is the layer that also has the task of ensuring that traffic padding mechanisms can safely be implemented. The SL identifies the various security functions that may be required in a connection and appropriately enforces them via the FCNS layers, maintaining at the same time the security contexts agreed between the peers.

The following sections contain details of the protocols and functions governing the FCNS realisation, while presenting each layer of the FCNS stack separately.

4.3 FCNS Functional View

FCNS layers' functions are realised by their respective protocols making use of the services provided by the layer below and so on. These presentations form the FCNS stack specification, to which the prototype and source code implementation had to conform.

FCNS has been designed as a secure framework into which various network protocols could be incorporated, for example by replacing one or more of its layers with an existing set of communication rules. To provide measurements and comparisons of the stack with other architectures, functionality has been implemented in each one of the FCNS layers, creating the necessary functions and services needed to establish a connection and, hence, measure the efficiency of FCNS under realistic operational conditions.

4.3.1 User-Defined Presentation (UDPRES) Layer

UDPRES forms the interface of the FCNS stack to the user initiating, maintaining and tearing down a particular connection. This user may be a UMTS node, or a host present on a packet-switched network topology, with the information to be carried varying from signalling to data files. UDPRES services are realised by the UDPRES protocol, which makes use of the services offered by the layer below (UDSES) to provide in turn facilities to the nodes requesting a connection. Its functionality can be summarised as follows:

- Interface between FCNS and applications
- Negotiation of presentation context / transfer syntax
- Representation of the abstract syntax requested by an application
- Maintenance of the information contents received
- Use of the UDSES services

The following two subsections illustrate how these functions are materialised in the FCNS stack and how they are translated into the services the UDPRES provides.

4.3.1.1 UDPRES Protocol Definition

This particular set of communication rules offers a confirmed service, meaning that it has to accept requests that are transformed to indications, and responses translated into confirmations. Figure A.3 of Appendix A illustrates the specification diagram of the UDPRES protocol and its functionality with respect to the services it provides to the nodes.

The requests are initiated, in FCNS terminology, by *decisions* and the responses by the *decision_responses* from the node instances. Table 4.1 depicts the various decisions and their responses used by the UDPRES protocol.

Depending on the decision and the respective response the UDPRES service translates them into the necessary service primitives, as illustrated in Table 4.2. In order to implement the UDPRES protocol into a functional architecture, service primitives are interpreted as messages exchanged between the FCNS layers, traversing from the source node to the ultimate destination and vice versa.

Table 4.1: Decisions and decision responses of the UDPRES protocol

Request for an action	Decision	Decision Response
Establish connection	<i>conn_request</i>	<i>conn_accept</i> / <i>conn_reject</i>
Data transfer	<i>data_request</i>	<i>data_accept</i> / <i>data_reject</i>
Finish data transmission	<i>data_finish</i>	<i>finish_accept</i> / <i>finish_reject</i>
Release connection	<i>release_request</i>	<i>release_accept</i> / <i>release_reject</i>
Error	<i>FATAL_error</i> / <i>NONFATAL_error</i> / <i>OTHER_error</i>	<i>error_FATAL</i> / <i>error_NONFATAL</i> / <i>error_OTHER</i>

Table 4.2: Service primitives exchanged by the UDPRES protocol

Action	Request	Indication	Response	Confirmation
Initialise	<i>connect_init</i>	<i>connect_init</i>	<i>con_setup</i>	<i>con_setup</i>
Data transfer	<i>data</i>	<i>data</i>	<i>data_transfer</i> / <i>engaged</i>	<i>data_transfer</i> / <i>engaged</i>
End transfer	<i>end_data</i>	<i>end_data</i>	<i>data_end</i>	<i>data_end</i>
Release	<i>disconnect</i>	<i>disconnect</i>	<i>con_exit</i>	<i>con_exit</i>
Error	<i>fault</i>	<i>fault</i>	<i>error_conn</i>	<i>error_conn</i>

The names of the *requests* and *indications* and those of the *responses* and *confirmations* respectively are the same for consistency reasons.

Ultimately UDPRES is the layer involved with arranging encoding issues for the data to be exchanged. The service primitives originated by this layer are used to negotiate the source and destination information representation formats, and agree upon the transit format, or *transfer syntax*, which will be used throughout the connection duration. This mechanism enables any higher layer applications to exploit the data communication control facilities offered by UDSES, by synchronising and maintaining a particular session. Its responsibilities though are limited to communicating and maintaining the set of syntaxes that the nodes require to be used. It is not involved in determining these and consequently its services to the nodes and network elements in a topology are the identification of the abstract syntaxes and their negotiation.

Once the connection establishment and data transfer phases have been completed, UDPRES must ensure that the agreed syntaxes are securely removed from the context lists present in the nodes, to avoid possible manipulation by an adversary. It is up to the UDPRES to ensure that the SL verifies, authenticates and secures the lists and messages exchanged between the peers, to minimise the possibility that an attacker could be successful in determining the type of syntax agreed for a connection.

4.3.1.2 UDPRES Service Definition

The UDPRES protocol has the task of presenting the FCNS stack with the requests and responses initiated by the network elements, in an attempt to establish and maintain a connection, providing the UDSES layer with the parameters necessary to ascertain a particular level of QoS in the session.

Its services can be summarised as follows:

- Identification of abstract syntaxes
- Selection of the appropriate transfer syntax for a connection
- Negotiation and maintenance of the agreed transfer syntax
- Maintenance of the QoS level provided by the UDSES protocol
- Access to the services provided by the UDSES protocol

As mentioned previously, these services are realised by the UDPRES protocol, in various phases of the communication, such as the connection establishment, data transfer and connection release.

4.3.1.2.1 UDPRES Connection Establishment

UDPRES connection establishment phase is initiated by a decision launched from the sending peer, indicating that information has to be transferred to a particular destination. Prior to identifying the abstract syntax requested by the application, the SL has to verify the validity of the node and arrange for the necessary security context to be agreed between the peers, enforcing the FCNS handshake mechanism used to enhance system's security (these actions are described in detail in Chapter 5).

Immediately after the reception of the initial decision from the node, the SL provides all FCNS layers with the necessary security algorithm parameters that will be used to secure the messages to be exchanged in the stack. The mechanism ensures that since there is no functional FCNS stack created for a connection, an adversary wishing to intercept these messages will be unsuccessful.

FCNS functionality is subject to the active participation of the node running its instance in a session, as is the case with every set of communication rules. UDPRES is in an idle state, until the node indicates its willingness to communicate

with a peer (sending instance), or receive information from one (receiving instance). If that is the case, SL instructs the UDPRES, together with the rest of the FCNS layers, as to the algorithms that will be used to secure the messages intended for the destination. If the process of accepting and verifying with the SL the required procedures is successfully completed, then UDPRES may change its status awaiting for the node to initiate its request. In any other case an alert is issued and the communication is halted.

Whenever a node issues the initial *conn_request* decision, UDPRES has to check whether it can first accept the request and then establish the status of the node with respect to any other activities it might be involved with. If the acceptability of the request cannot be achieved, UDPRES issues an alert to notify the application and the SL that the particular request cannot be fulfilled. On the contrary, if UDPRES can process it, it transforms the decision to the *connect_init* request, which is sent to the SL via the *CONNECT_REQUEST* message. The SL initiates a check sequence for the request and starts the handshake mechanism used to establish a secured communication path between peer nodes and the Security Contexts (SCs) to be used throughout the session.

At the receiving end, UDPRES has the task of identifying the service primitive that has arrived and check whether it can be accepted according to the current phase of the communication. If the request cannot be met then an alert is sent to indicate the error. On the other hand, if the request is validated then it is transformed to an indication and sent to the receiving process.

Upon reception of the confirmation for the initial request, UDPRES has the task of establishing the syntax that will be used for the data that needs to be transmitted, hence moving into the context setup and management phase.

4.3.1.2.2 UDPRES Context Setup and Management

Context setup and management functions are involved in negotiating and preserving the presentation context agreed between peers for the connection. There may be a case where the sending or receiving entities wish to alter this context, something that has to be supported by UDPRES, to enable communication to proceed without releasing it or causing any service discontinuity. For the negotiation of the transfer syntax between peers, UDPRES uses the NODE_REQUEST, NODE_REQUEST_RESPONSE and NODE_ESTABLISH messages, depicted in Figures 4.5, 4.6 and 4.7 respectively.

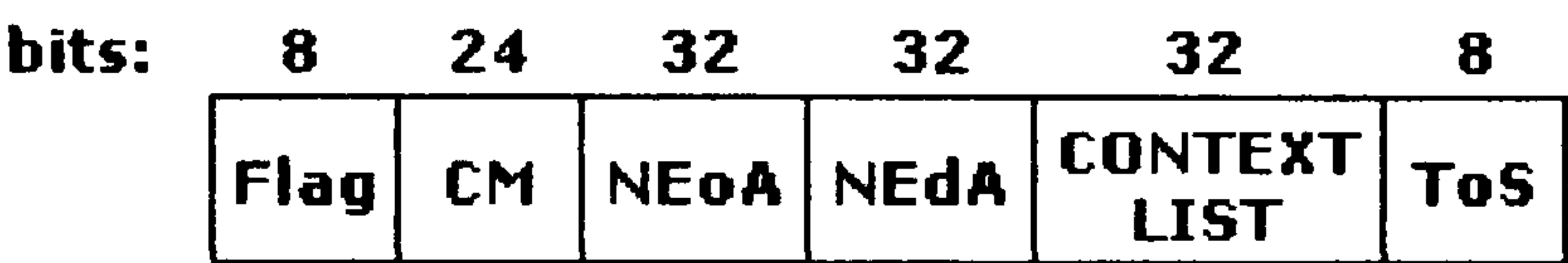


Figure 4.5: NODE_REQUEST message

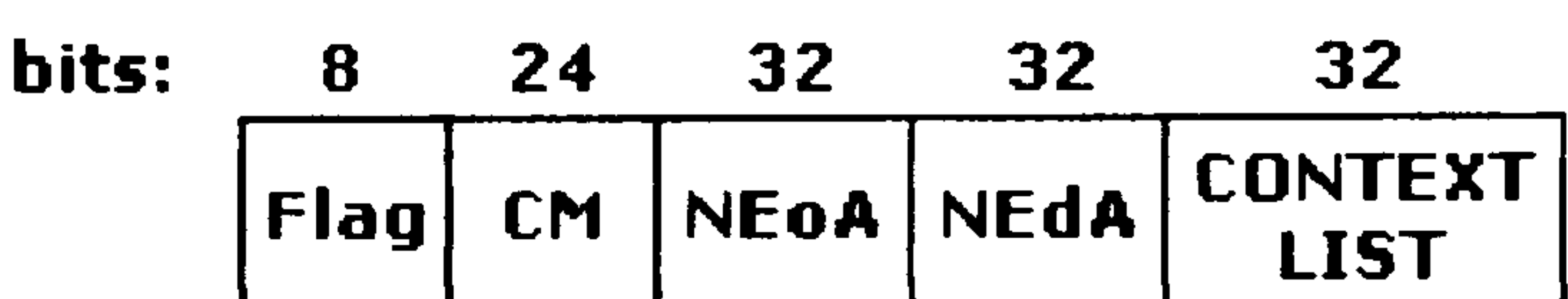


Figure 4.6: NODE_REQUEST_RESPONSE message

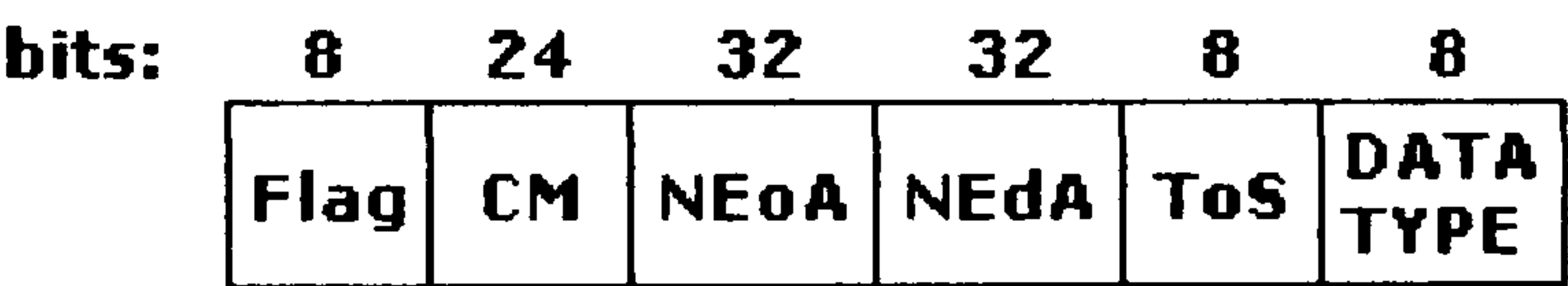


Figure 4.7: NODE_ESTABLISH message

The *flag* field is a variable used in the simulation environment for measurement purposes and also to enable nodes to identify and discard any duplicate messages. The *CM* field contains information about the security of the particular message agreed during the SC exchange phase, such as authentication and confidentiality identifiers. Additionally, the *NEoA* (Network Element originating Address) and *NEdA* (Network Element destination Address) fields contain the sending and receiving addresses, whereas the *ToS* field of the NODE_REQUEST and NODE_ESTABLISH messages includes information as to the type of service that is requested by the application. Finally, the *CONTEXT LIST* field represents the list of the abstract syntaxes that will be used for the agreement of the connection transfer syntax, whilst the *DATA TYPE* field indicates whether the information to be exchanged is user or signalling data.

The negotiation of the syntax is done via the *NODE_REQUEST* message, where the source indicates the most convenient context for it, as well as the type of service and data to be in transit. The receiver then initiates a check sequence to identify whether it can support the requested syntax and type of service for the connection and includes its response at the *NODE_REQUEST_RESPONSE* message. If the request from the sender cannot be met, then the process continues so that peers can come to an agreement of a syntax convenient for both nodes, which will be used and preserved throughout the session. This process usually takes one step in FCNS, since the initial *NODE_REQUEST_RESPONSE* message would include information about a syntax that the sender could support, selected from a context definition list residing in the original message and found at the receiving entity. If there is no agreement though, then a second *NODE_REQUEST* message is sent to the receiver, which has to accept the parameters indicated by the source, else the connection establishment procedure is halted.

Success in setting up the transfer syntax for the session (presentation context) is indicated by the *NODE_ESTABLISH* message, which includes the type of service and type of data that is to be transferred throughout the data transfer phase.

There may also be cases where the presentation context must be changed during the course of a session, for example due to a software fault in one of the peers, leading to misidentification of a transfer syntax. Whenever such an issue arises, the *NODE_REQUEST* message is issued to indicate that one or many presentation contexts are to be added or deleted from the existing list, or that the transfer syntax must be changed for that session. The indication for such an action is included in the data messages sent to the receiver, prior to releasing the *NODE_REQUEST* message, to ensure that the request has indeed been issued by the intended peer, and is not part of any insertion or Denial of Service (DoS) attack. The procedure in arranging the alteration of the context is similar to the one used in the connection establishment phase, though data transfer resumes

without the nodes having to issue a *data_request* decision. If the context and the requested action are not accepted by the peer, then an alert is issued to the SL to indicate the type of error which has occurred.

Depending on the network running on top of FCNS, context exchange times may vary. For the UMTS CN, context agreement should not take more than 100 msec [142], a point of measure set for the FCNS implementation and simulation environment.

4.3.1.2.3 UDPRES Data Transfer

UDPRES offers information transfer between peer nodes depending on the settings agreed between them in the connection establishment phase.

The sending node issues the appropriate decision, which is transformed to the respective request and sent to the receiver to check whether it can accept the information or not. In case the request is rejected, the sender will have to wait for a predetermined period of time before issuing the same request. If the receiver rejects the request more than 3 times, then the communication is dropped. In any other case, the *data_accept* decision response is interpreted in the respective response primitive and sent to the sending instance.

Information is passed from the sending process to UDPRES, which first performs a check to verify that the syntax conforms to that agreed between the nodes. In case of a match, UDPRES adds its own header to the message passed to it, making sure that it is encrypted with the keys and algorithms arranged for this particular connection. Data messages are forwarded to UDSES in the form of the *NODE_TRANSFER* message, depicted in Figure 4.8, to make the layer below aware of the current phase of the session. If UDSES cannot accept the message, it sends

an unacknowledgement back to UDPRES to indicate the reason why it has rejected the message, and the time UDPRES should wait before attempting to send it again.

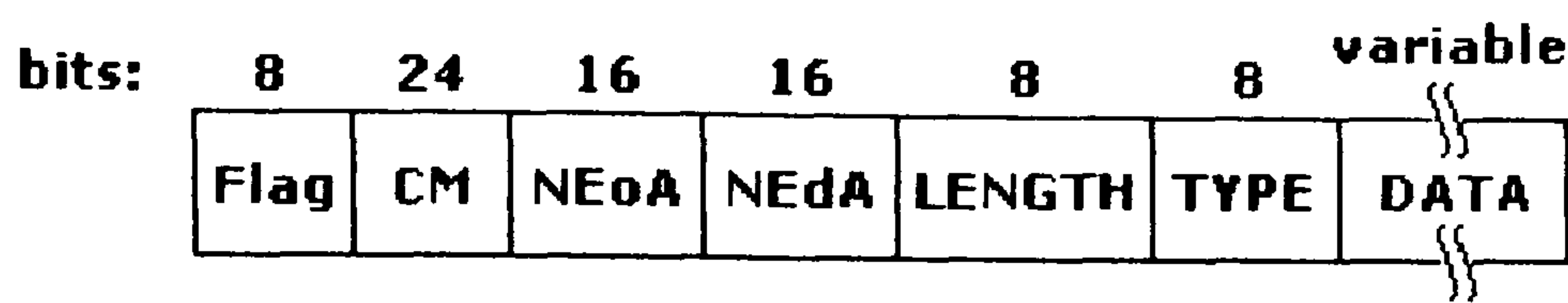


Figure 4.8: UDPRES NODE_TRANSFER message

The *LENGTH* field indicates the length of the whole message, whereas the *TYPE* field includes information as to whether the NODE_TRANSFER is used to transport user or control information.

At the receiving end, UDPRES ensures that the representation of the information received is correct, verifies with the SL that the algorithm and secret keys used to encrypt the data conform to the parameters set, and decrypts the message arrived from UDSES. If decryption and message authentication are completed successfully, then UDPRES strips its header and passes the information to the receiver, waiting for the next message to arrive.

For any errors occurring due to internal faults, such as erroneous actions by the FCNS layers, the message sent to initiate the FCNS Error Protocol (FCNSEP) is the *NODE_ALERT* message illustrated in Figure 4.9, whereas in cases where communication must be ended the *NODE_ABORT* message of Figure 4.10 is used.

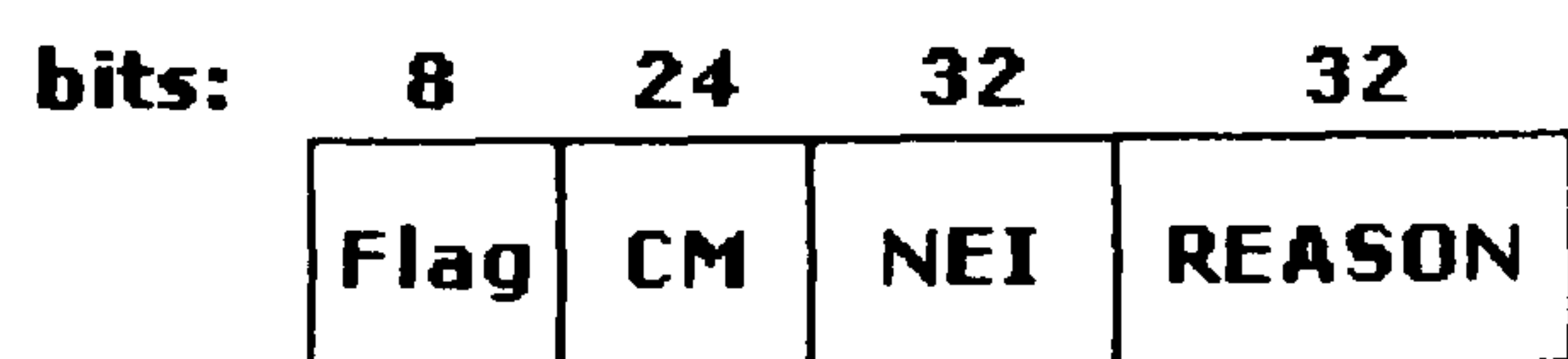


Figure 4.9: NODE_ALERT message

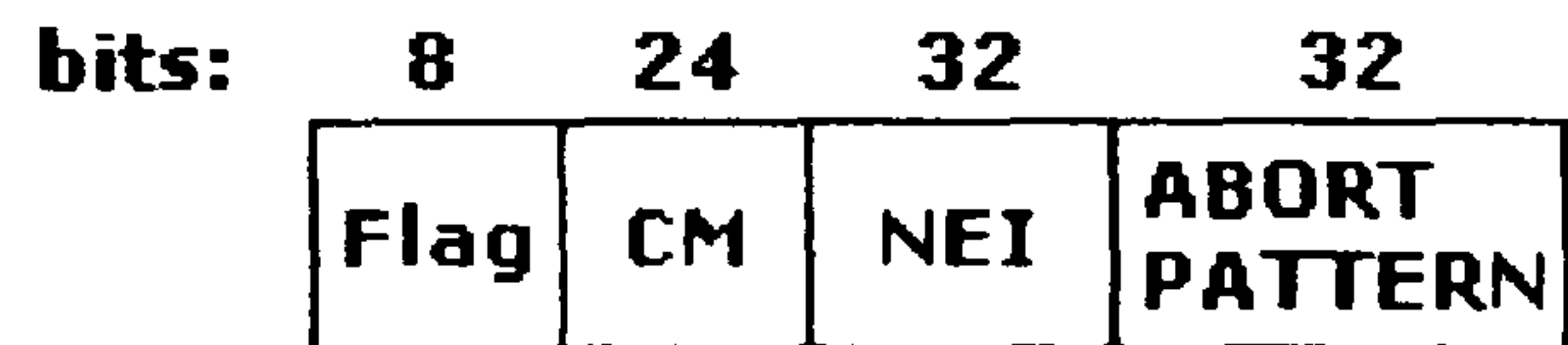


Figure 4.10: NODE_ABORT message

In the *NODE_ALERT* message the *REASON* field includes information as to the type of error that occurred, whilst the *NEI* (Network Element Identifier) field indicates the FCNS layer that detected the error. On the other hand, the *ABORT PATTERN*

field contains a series of bits initiating the decision of aborting the communication from the particular layer's point of view. It is up to the SL to check the type of error and decide whether to initialise the FCNSEP and also whether communication should indeed finish or proceed.

Finally, whenever a node finishes sending information, it sends the *data_finish* decision via UDPRES to the peer. Rejection of the request is indicated by the *finish_reject* response, issued in case there is more data to be received, a situation that may occur if some of the data messages have been routed via a different link due to congestion or transmission error and did not manage to reach their destination on time. On the contrary, *finish_accept* denotes a decision acceptance and also that the connection release phase should normally follow.

4.3.1.2.4 UDPRES Connection Release

There are two types of connection releases for the UDPRES, as with every other FCNS layer. These are the orderly and disorderly release that may be initiated by any peer throughout a communication phase.

Orderly connection termination takes place whenever a node has completed sending or receiving information. The node issues a *release_request* decision, which is conveyed to the necessary request primitive via the *NODE_END* message depicted in Figure 4.11. UDPRES sends the response back to source going into the final stages of completing the session. If the response was negative to the request (*release_reject*) then normally another *NODE_END* message must arrive. If this is not the case, an alert is issued to the source and the SL indicating that the connection cannot be ended normally. On the other hand, if the release is accepted, the source issues the final *NODE_IDLE* message to indicate that its status has gone back to the *idle* state. UDPRES status may still remain in a working state,

since it may be involved with other activities, but for the particular application, it has to become idle.

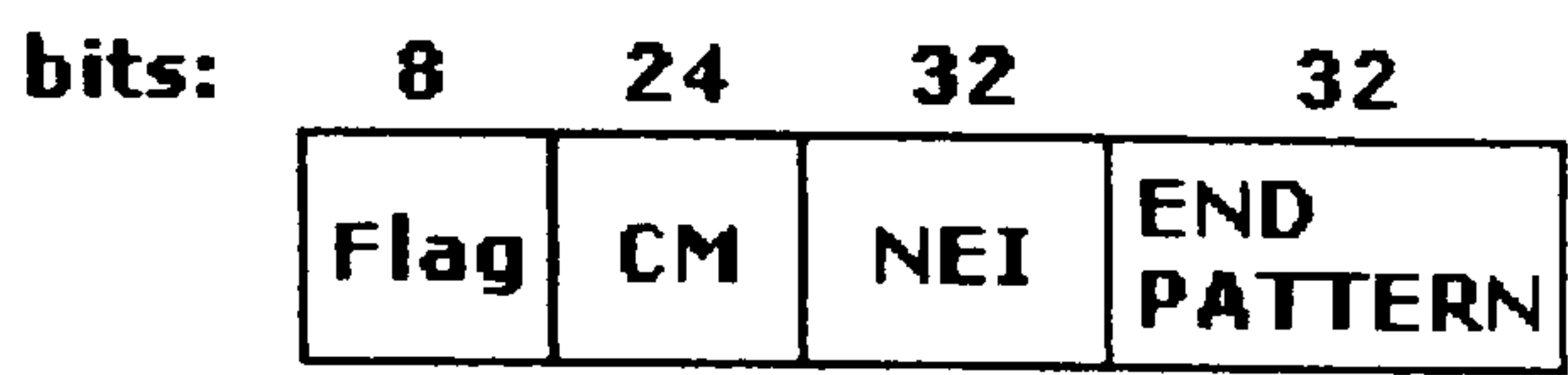


Figure 4.11: NODE_END message

In the NODE_END message the *END_PATTERN* field contains a bit series indicating that communication may be terminated.

Disorderly release results from a *FATAL* alert issue, which is interpreted by the *COMM_ABORT* message, which is similar to the NODE_ABORT given in Figure 4.10. Upon reception of such a message, the SL verifies that indeed there is such an error for which communication must end, prior to instructing the UDPRES to go back to the idle state, closing down any pending sessions from the particular node regardless of their phases. For communication to proceed, connection establishment must take place again, renewing the algorithms, secret keys and context list for security purposes.

4.3.2 User-Defined Session (UDSES) Layer

The UDSES layer provides the services and functions of the UDSES protocol, which is mainly involved with synchronisation issues and address verification of the peers involved in a session. UDSES handles the data that is passed to it by UDPRES, enhancing the end-to-end transfer services offered by the TX_LAYER, by managing the data transfer phase over a session.

Since for the FCNS realisation the nodes have been assumed to be peers, UDSES does not provide any token service that could be used in initiating the release of a connection, although support for such services is included in the protocol

specification. Any user may terminate the session at any time, either using the orderly or disorderly release service of UDSES. Similarly, every node can transmit or receive information at will, given that UDSES is able to support this action at that time.

The functions of the UDSES protocol can be summarised as follows:

- Full duplex communication support
- Address verification functions (in conjunction with the SL)
- Negotiation and maintenance of synchronisation issues
- Control of connection QoS parameters
- Negotiation and maintenance of segmentation/reassembly and concatenation/separation parameters
- Use of the TX_LAYER services for the UDSES user

UDSES functions are realised via the UDSES protocol, which makes use of the services provided by TX_LAYER, to afford access to the services required by its user, that is the UDPRES layer. The following subsections illustrate how these functions are realised in the FCNS architecture leading to the provision of the UDSES services.

4.3.2.1 UDSES Protocol Definition

The UDSES protocol is responsible for providing synchronised communication between peers, offering an agreed QoS for the connection and maintaining that level throughout its duration. Figure A.4 of Appendix A depicts an abstraction of the UDSES protocol specification diagram and its functionality.

UDSES functions come into effect whenever UDPRES issues the connection request for a particular destination. The protocol enters a new phase where it has to negotiate with the peer the required QoS parameters that the sending instance has

indicated, making use of the available resources and services that are offered by the TX_LAYER. If an agreement cannot be achieved, then the SL seizes communication and an alert is sent to indicate the error.

Sending and receiving information to and from UDSES users though, is a process that has to follow an essential address verification procedure. Since there may be malicious users and hosts present in a topology, there must be a certain level of assurance that the connection has not been initiated or received by an unauthorised entity. This feature works in conjunction with data origin authentication issues, though at the very first phases of establishing a communication, if a malicious user could impersonate a valid node then that could prove any security measures ineffectual. An example of this case would be an illicit user that uses its address to initiate a session with a legitimate node. If the attacker is able to get hold of the SCs exchanged between the peers, then it becomes able to perform any action within a session it sets up, since it holds the necessary parameters to masquerade as a valid node. With the address verification procedure there is the extra measure that, given that the address databases are adequately secured, the messages sent and received by the FCNS stack are continuously checked for inconsistencies. Even if the security context is compromised, then there is a process that attempts to identify any impersonation by the attacker. Although such a measure cannot guarantee that address-spoofing attempts would be unsuccessful, with the network databases being secured, FCNS should be able to detect any address unknown to it and issue an alert to make such a case known to the operator.

Additionally, UDSES has to ensure that the correct synchronisation parameters will be exchanged and controlled between the peers. It may be the case, for example, where the sending instance is able to send information at greater rates than those at which the destination can receive it or vice versa. There must always be an agreement between the peers, to avoid congestion and blocking, and at the same time prevent an adversary taking advantage of either of these two cases for a DoS

attack. Furthermore, since the communication supported by the UDSES is full duplex, synchronisation is used to regulate traffic flow in the network, adding to the explicit flow control implemented in the TX_LAYER.

As UDSES makes use of the services provided by the TX_LAYER to support the requested QoS for a session, it is the layer where the control of such parameters takes place, so that the application can be aware of the TX_LAYER capabilities in supporting and maintaining the required levels. Whenever there is an indication that the agreed QoS levels are not enforced, UDSES issues an alert to indicate the error. If such errors persist then communication must be ended, since these could be the result of a software or hardware fault at the nodes, or due to an adversary trying to interfere with the connection.

Segmentation and reassembly are also issues that can be exploited in the UDSES protocol, since they are related to the way UDSES messages can be transferred between peer entities. These messages that are initiated by the UDSES user, form the Service Data Units (SDUs) of UDSES, which are passed to the TX_LAYER as the UDSES Protocol Data Units (PDUs). If segmentation is selected, then a single UDSES SDU could be sent to the peer entity in segments, that is, multiple UDSES PDUs. This selection could take place as a result of the user data being of a size bigger than that set by the Maximum Transfer Unit (MTU) of the particular network topology. At the receiving end, the receiver would reassemble these segments to obtain the full UDSES message. This procedure also takes place at the TX_LAYER entities, where there is a better understanding of the network capabilities, since TX_LAYER sits on top of the subnetwork FCNS runs on. In UDSES, segmentation could also be initiated as a result of a limited buffer space, or limited file handling capacity, or by large service primitive messages.

Additional functions of the UDSES protocol include concatenation and separation functions that can be used to improve the efficiency of a system in case various

small segments are sent from the UDSES users. The UDSES PDUs can be mapped into one single TX_LAYER SDU and sent to the peer entity, where they are separated to provide the UDSES entity with the appropriate segments. Such a case may arise as a result of the transfer of signalling data in a network where MTU sizes can be quite large compared to the size of the data that is to be transferred. By mapping several UDSES PDUs into a single TX_LAYER SDU will increase the performance of the system in two ways. Firstly, fewer packets on the links decrease the probability of blocking and, most importantly, the probability of segment loss that could result in retransmission. Since the packets are kept in a buffer until their reception is acknowledged, a packet loss implies retransmission of only that particular packet, which includes the number of segments mapped into it. Secondly, it decreases the time taken for TX_LAYER PDUs to be secured and exchanged between the peers, since fewer packets means less processing load at both the sender and the receiver arising from the encryption/decryption processes.

The segmentation/reassembly and concatenation/separation functions have been included in the FCNS design to test the performance of the stack with regards to the impact such functions would have in the overall security performance of the system. They are negotiated and managed by the UDSES protocol and they are visible only to the peer UDSES entities, not the UDPRES.

4.3.2.2 UDSES Service Definition

The UDSES protocol has the task of presenting UDPRES with the services offered by the TX_LAYER, to implement and maintain the required QoS parameters in the connection. The functions present in UDSES have the objective of enforcing the services provided by the protocol, which can be summarised as follows:

- Session establishment and release
- Data transfer
- Session management

- Session synchronisation and QoS negotiation

The session synchronisation and QoS negotiation services form part of the session establishment and management phase, since the parameters agreed are essential for the setting up and control of a particular session. Synchronisation management and QoS management do not explicitly form phases of the UDSES protocol. They are included in the FCNS design to enable the use of these services whenever needed and are both treated in this chapter as individual segments of the session management phase for illustrative purposes.

4.3.2.2.1 UDSES Session Establishment

As with every FCNS layer, the session establishment phase takes place after the SL has instructed the UDSES layer of the algorithms and secret keys it can use to secure the messages that are to be exchanged with the peer entity. These instructions by the SL are mandatory and communication cannot proceed unless the required security phases have been completed. Failure in doing so will result in the connection being halted and the issuing of an alert to indicate a severe fault of the FCNS stack initialisation.

Whenever a connection request arrives from the UDPRES layer, UDSES checks whether it can accept it and set up the session for the particular application instance. The address of the sending entity is checked to verify its validity against the list of valid nodes for the network topology. Rejecting a particular address may be the result of an invalid sending instance, an invalid receiver address or both.

If the validity of both nodes is verified, then the request sent by the UDPRES can be analysed and the appropriate actions can be taken for the session to be established. These actions include the negotiation of the synchronisation points

and parameters between the UDSES peers and the agreement on the required QoS levels, as set by the underlying TX_LAYER.

The mechanism is triggered upon reception of the NODE_ESTABLISH message, which indicates that the presentation context for the connection has been agreed between the nodes. UDSES initiates the routine for establishing the necessary criteria that will be used to set the correct QoS features for the session. Table 4.3 indicates the possible parameters that user could request for a connection.

Table 4.3: FCNS QoS parameters

Parameter	Value
QoS level of support	<ul style="list-style-type: none">- Maximum level: No residual errors accepted, throughput above 70%, no transit delay, high priority- Medium level: No residual errors accepted, throughput above 50%, transit delay below t msec, high priority- Low Level: Residual errors accepted, throughput above 50%, transit delay below 2t msec, low priority- Zero Level: Residual errors accepted, no throughput restrictions, transit delay below t msec, no priority restriction
Segmentation support	<ul style="list-style-type: none">- No length restriction, segmentation not required- MTU can be 1500 bytes long- MTU lower than 1500 bytes, of size x bytes
Concatenation support	<ul style="list-style-type: none">- No concatenation can be accepted- Concatenation may take place
Service access level	<ul style="list-style-type: none">- Full service access (only case supported in FCNS)

As seen in Table 4.3 the UDSES user can choose between several QoS levels varying from high quality communication requirements to low quality ones. For the FCNS simulation environment, the stack has been put under test with maximum and medium QoS levels to test its performance in cases where transit delay could not be acceptable, or at least at a certain level. The QoS level may change

throughout the session provided that the change passes through the SL procedures, to ensure that the alteration does not form part of an attack by an unauthorised user.

Once the QoS values have been set with the application, UDSES has the task of notifying the TX_LAYER of the decisions taken to ensure their support throughout the connection. If the requirements cannot be met, then UDSES informs the application that the selected parameters need to be changed and the process may be repeated only once more provided that UDSES informs the node of the support TX_LAYER can offer. If the second attempt is unsuccessful too, then the error is deemed fatal and communication is aborted. On the other hand, if the application requirements can be met, UDSES issues the required permission to the node to start transmitting the data.

At the UDSES layer, synchronisation encompasses issues involving the instructions UDSES makes to the TX_LAYER, mainly concerning the time at which TX_LAYER transforms the UDSES PDUs into the messages that are to be sent. The process is usually triggered by either one of the peers present in the session, and aims at regulating the flow of data in the network, as a result of the UDSES synchronising the data flow. Since synchronisation takes place throughout the whole of the communication phases, it is discussed in the subsection concerning the synchronisation and QoS management.

4.3.2.2.2 UDSES Data Transfer

UDSES enters the data transfer phase immediately following the issuing of the permission to the sending and receiving instances for the beginning of the data exchange. This permission is actually a means of verifying that the session establishment phase has been completed as expected and that all the necessary parameters and variables have been set as required.

UDSES makes a connection request to the TX_LAYER to check the availability and status of the layer prior to forwarding the data. Failure in obtaining a valid status response, after a maximum of three consecutive tries from the layer below, implies the termination of the connection. On the contrary, if TX_LAYER can accept the information, then UDSSES accepts the data passed onto it by the UDPRES and forwards it to the TX_LAYER in the form of the *MSG_CREATE* message, depicted in Figure 4.12.

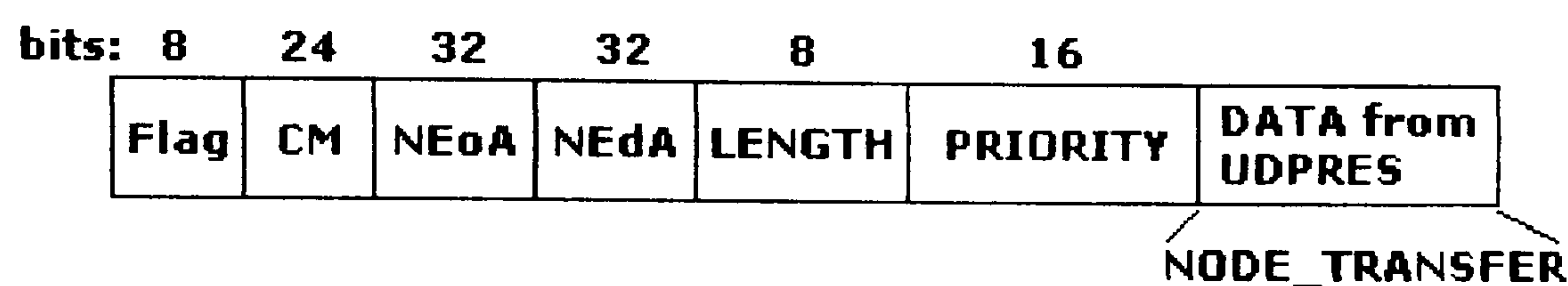


Figure 4.12: UDSSES MSG_CREATE message

The NEoA and NEdA fields indicate the sending and receiving instances of the UDSSES protocol that the message has been created and is intended for respectively. The *LENGTH* field indicates the length of the whole of the message in bytes, whereas the *PRIORITY* field is a measure given to the TX_LAYER as to the priority level at which message should be treated.

Upon reception of the MSG_CREATE message by the receiving UDSSES entity, the address verification process is initiated, to check the validity of the addresses present in the received data. If the addresses are validated, then the message is stripped of its header and is passed to the UDPRES layer, waiting for the next message to arrive, provided that the last one received did not have the End Of File (EOF) identifier. On the other hand, if even one of the addresses is proven to be invalid, then the message is dropped and an alert is issued upon persistency of the error. Usually, such a case may arise by finding the sending node to be an invalid one. That does not imply though that the address present in the *NEdA* field would contain a valid address identifier. Although it is up to the PHYS and EE_LAYER to forward the packets to their correct destination, it may be the case that a message intended for another node has somehow reached the node actively participating in

this session. For the UDSES layer this is regarded as an error and the reception of three such messages results in a FATAL error report and consequently the disorderly release of the connection.

If the sending application instance finishes transmitting the data, then the UDPRES notifies the layer of its intention of releasing the session and terminating any outstanding processes for this particular connection.

4.3.2.2.3 UDSES Session Release

UDSES session release phase follows similar principles to that of the UDPRES connection release in the sense that it provides both the orderly and disorderly session termination services to its user.

Orderly release follows the appropriate request made by the UDPRES protocol in the form of the NODE_END message. The layer has the task of identifying whether there any outstanding activities in the session for which the request has been made, before proceeding into terminating the connection. If there are any pending service actions, UDSES has to reject the request, so that these processes can finish, prior to resetting all parameter values set for the connection. Such a case may occur due to an error made by the application assuming that data transfer has normally finished before accepting any acknowledgement for the last message sent. Furthermore, an adversary could try to force the tearing down of the session by inserting bogus NODE_END messages into the traffic flow. It is up to the UDSES layer to ensure that, in conjunction with the address verification procedure, the connection cannot be released until all session processes have come to an end.

The reception of the NODE_END message by both the sending and receiving UDSES peer entities forces the protocol into entering a waiting state, whereby the SL has

to issue a command before UDSES can go back to its initial idle condition. This is achieved by using the *SET_STATUS* message as depicted in Figure 4.13.

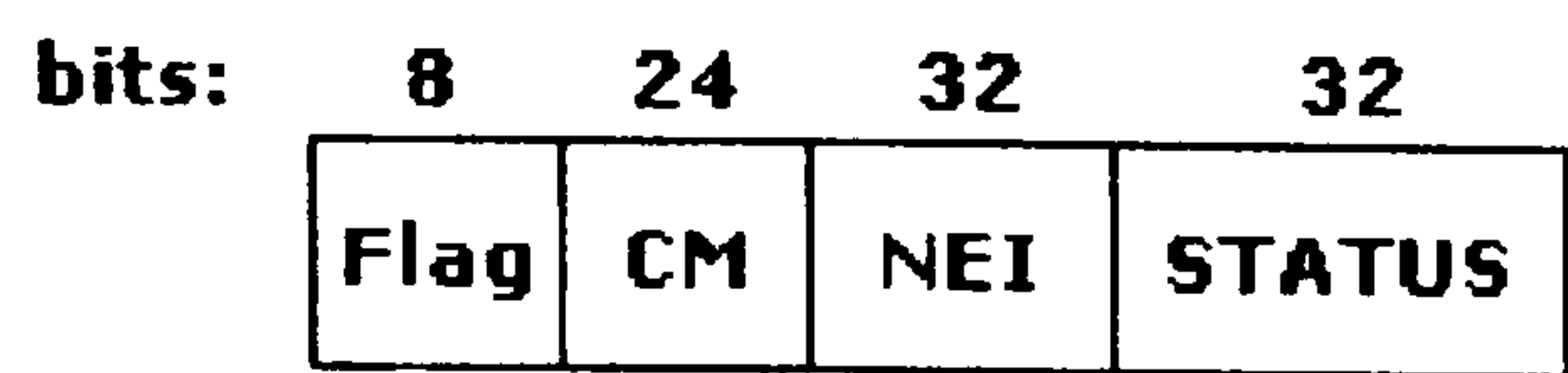


Figure 4.13: *SET_STATUS* message

The *STATUS* field of the message contains a series of bits indicating that the layer should go into an idle state, since its particular instance is not needed in the FCNS stack at that time. Although UDSES may be used moments after the release of a session to establish a new one, it is imperative that it returns to an initial idle state, as is the case with all the FCNS layers, to avoid livelocks and deadlocks in the protocol implementation. The process ensures that all parameters used throughout the connection are reset and that the protocol temporarily loses its functionality to avoid attacks aiming at the source code used to provide it. Lack of protocol functionality implies that an attacker would not be able to launch an attack at the protocol, simply because no instance of it would exist.

Disorderly release in the UDSES protocol may occur for a variety of reasons, including routine failures, bad encryption/decryption processes, lack of negotiation successions and the like. Whenever such an error, classified in FCNS as fatal, occurs, the *COMM_ABORT* message is used to indicate the immediate termination of the connection. This tearing down though has to be first validated by the SL, to protect the system from possible DoS attacks, aiming at connection discontinuity. Following the reception of such a message, the UDSES layer goes into an idle state, resetting all parameters and QoS values set for the connection. All nodes participating in a connection may break the communication with the peer, if the *COMM_ABORT* message is issued.

4.3.2.2.4 UDSES QoS and Synchronisation Management

QoS and synchronisation management are services provided to the UDSES user throughout the duration of the session and fall into the session management services category provided by the UDSES protocol. They can be initiated at any time from a peer UDSES entity and are used to enable users to comply with changes that the network topology may endure during the data transfer phase. An example of such a case would be the blocking or congestion built up in a particular link, where time-sensitive data is in transit. The peer processes may then choose to alter some QoS parameters, like for example tolerance in the transit delay, and hence have their packets routed via a different link. Similarly, the receiving entity due to a possible involvement in several activities may not be able to cope with the sending rate of the messages after a period of time already in the session. It is up to the UDSES protocol to facilitate the renegotiation of synchronisation parameters to increase the efficiency of the protocol and its adaptability in network changes.

If the QoS parameters need to be changed, then the node indicates this by inserting in the data flow a normal data message containing a parameter signifying that the message to follow does not contain user but control information. Upon acceptance of this message the receiver will have to put the messages already received in a buffer, and acknowledge any outstanding messages on hold. The sending instance then issues the request to the UDSES protocol to depict the values that need to be changed. Upon reception of the message, UDSES has to negotiate those with the peer entity, as well as with the TX_LAYER. If negotiation cannot be established, then the communication is terminated since it cannot further proceed. On the other hand, success in arranging the new QoS levels will result in the UDSES layer issuing permission to the peer nodes, signifying that communication should continue on the new basis set for that session.

Synchronisation management is achieved by monitoring the response of the peer entities in sending and receiving the information in transit. Upon reception of the

service primitive message, UDSES starts a check routine to establish whether the request has been significant, meaning that parameters should actually change for the continuation of the session and the request is not a product of a possible connection manipulation attempt by an adversary. If synchronisation has to take place, then the necessary adjustments are made and signalled to the SL. Communication can then proceed as normal, yet there cannot be more than three requests for synchronisation made by the same peers. Such a situation would imply the immediate abortion of the connection and signalling of the error to the system.

4.3.3 Transmission Layer (TX_LAYER)

Transmission layer is the FCNS layer responsible for providing the UDSES protocol the services offered by the EE_LAYER. It handles the data that is passed to it by UDSES, realising the end-to-end data transmission service of the FCNS, taking advantage of the link-based transmission nature of the EE_LAYER.

TX_LAYER functions can be summarised as follows:

- Provision of end-to-end transmission services
- Setup, control and release of connections
- Data transfer
- Multiplexing of TX_LAYER connections onto an EE_LAYER connection
- Datagram segmentation/reassembly and concatenation/separation
- End-to-end error detection and recovery
- QoS management
- Support for synchronisation services
- End-to-end security services
- Explicit flow control
- Use of the EE_LAYER services

The TX_LAYER functions are fulfilled by the TX_LAYER protocol, which is mainly involved with the establishment of secured end-to-end transmission connections between the peers. Figure A.5 of Appendix A illustrates the specification diagram of the TX_LAYER protocol, in relation to the layers adjacent to it.

4.3.3.1 TX_LAYER Protocol Definition

The TX_LAYER protocol forms the interface between the FCNS upper layers and those involved with the realisation of the subnetwork on which FCNS runs. Its involvement in the overall communication is essential for the maintenance and support of a certain quality of service requested by the user, incorporating security and error detection functionality.

End-to-end based communication implies the transmission of the data from the sending instance to the receiver, irrespective of the underlying subnetwork and the number of intermediate nodes the messages could be routed through. TX_LAYER entities recognise only the end addresses of the peers involved in the communication and it is on this basis that the protocol services can be deployed. Communication support in FCNS is full duplex and the TX_LAYER has been designed so that its instances running on the peers can act both as senders and receivers. The end-to-end nature of the protocol is achieved throughout the transmission connection phases as set out in the TX_LAYER Service Definition section.

To facilitate and enhance the services the TX_LAYER provides to the UDSES layer, it employs QoS and synchronisation management functions. These aim at maintaining the quality of the connection according to the levels agreed between the peers during the connection establishment phase. Nodes participating in a transmission may request the alteration or renegotiation of the QoS parameters, in cases where these cannot be supported for a particular reason, as described in the UDSES service specification. The TX_LAYER has to ensure that it can sustain the

agreement of the new values, without the process affecting the overall quality of the connection. Such processes may take place throughout all phases of the transmission, except its disorderly release, where communication is immediately broken.

Re-synchronisation of the connection may also be required in cases where sender and receiver rates deviate from the ones agreed between the TX_LAYER users, due to possible hardware faults in the nodes affecting the realisation of the FCNS instances, or attacks launched by an adversary aiming at connection manipulation. The TX_LAYER needs to ensure that the service is supported between the peer UDSES instances, without making the nodes aware of the situation that has arisen. In such an occurrence, the TX_LAYER shall not discontinue the normal data flow of the network instead it has to provide the necessary messages to the node instances alongside the data messages (datagrams). It will then be up to the connection users to comply with the new values and parameters set by the TX_LAYER for the transmission to proceed as normal. Failure in doing so will result in the connection's immediate release and the notification of the SL of the error situation. An exception of this case may occur due to an error detected only by the underlying subnetwork. If the error is not signalled to the TX_LAYER, then the protocol may not be able to recover from the situation and hence arrange for the new parameters to be agreed between the peers.

Moreover, since the FCNS Transmission layer has been designed to promote the end-to-end connectivity of the network topology nodes, it is the layer where functions such as end-to-end error detection and recovery are afforded. Usually the errors residual to the TX_LAYER instance concern the flow of the data between the ultimate peers or other problems that may have arisen from the transmission of the data messages. These may include timer expirations whilst waiting for the data acknowledgements, messages not being secured prior to their exchange, corrupted datagram headers, miscalculated checksums, corrupted or duplicate messages,

invalid end-to-end address identifiers and datagram sizes outside the length restrictions imposed during connection setup.

To compensate for corrupted received messages, the TX_LAYER uses a checksum value calculated on the whole of the datagram. At the receiver, immediately after the decryption and authentication of the datagram, the TX_LAYER instance recalculates the checksum value for the message received. If the value matches the one received then the message is stripped off its header and is then passed onto the UDSES layer. On the other hand, if the two values differ, then the message is dropped. To protect against message corruption by either a transmission link error or an attacker launching a modification attack, the TX_LAYER notifies the SL of the situation after receiving more than three such messages. The SL has the task of informing the system of the situation and establishing new parameters for the connection, such as the transmission of the datagrams via a different route. Persistency of such an error will result in the communication being seized by the SL to protect the protocol stack mechanisms against possible attacks.

More severe error cases include the transmission of datagrams in plaintext form, even when the end-to-end security functions are in effect. These types of errors can be the result of either a software fault in the FCNS implementation at the specific node originating the situation, or due to an attack aiming at manipulating the FCNS implementation on that particular node. These errors cannot be signalled by the subnetwork since the EE_LAYER cannot process the datagram headers to check their validity. If the receiving TX_LAYER entity detects such an error, then the SL is given notice of the situation so that it issues an alert via the FCNSEP to the sending entity. The connection is then entered a new mode, whereby all messages intended for the particular destination are explicitly monitored by the SL, prior to their forwarding to the TX_LAYER. Communication is disorderly released if the TX_LAYER is not able to recover from the error occurred and the node in

question is put into a suspension mode from future connection requests, until the situation is cleared by the network operator.

Other residual error circumstances like invalid end-to-end address identifiers are usually treated as a result of a fault in the transmission link. Although an aggressor could launch a modification attack for various reasons including the denial of the transmission service, it is not very likely that modifying the end-to-end address identifiers will have great effect as to the success of the attack. Severe problems could also arise by an adversary aiming at changing the security fields of the message in such a manner that he/she could impersonate a valid node and manage to get the modified messages through. Yet, FCNS offers several degrees of protection in all levels of the communication, making the chances of success for such attacks very low. In any case, if the TX_LAYER instance originating the error cannot recover from it, then the SL has to seize the communication. It is recommended that the connection be routed via another node or serving environment, so that the service can still be offered to the user.

Another category of residual errors includes situations where acknowledgements for the datagrams sent do not arrive in a predetermined period of time. This may be the result of faulty or congested network links, or a software fault on the node running the TX_LAYER instances. In the first case, the FCNSEP is used to indicate to the sender that timers should be altered to compensate for the time it would take to either solve the problem or route the datagrams via another channel. In the second case, the problem may be a severe one in the sense that even if such a suggestion is made, the TX_LAYER instance may not be able to adjust to the new situation. Such errors fall outside the scope of the FCNSEP and error detection and correction mechanisms, given that any source code implementation failure would affect the stack as a whole.

Finally, TX_LAYER errors may be the result of datagrams arriving with length larger than that agreed between peers in the connection establishment phase. This error is quite significant since its existence indicates that either it is the consequence of a software fault, or a modification attack. Since the parameters governing the size of the datagram to be exchanged are decided and verified prior to the data transfer phase (QoS negotiation), an error of this nature signifies the use of the FCNSEP and the explicit monitoring of the connection for recovery purposes.

Flow control in the TX_LAYER is achieved by using a mechanism similar to the sliding window algorithm of the TCP/IP architecture [28]. This method is used to provide basic reliable message transport, however many different techniques can be implemented to enhance it further. In FCNS, the TX_LAYER obtains the necessary information to enforce the flow control mechanism by consulting its users regarding their requirements and their ability to support the traffic flow once established. The mechanism in this layer is explicitly enforced, meaning that flow control is implemented irrespective of the underlying subnetwork and the technology supporting it. The intention of the TX_LAYER is the ordered and in-sequence delivery of the datagrams between the peers, no matter the route they will traverse to reach their destination.

The sender must first be able to adjust its sending rate to the respective one of the receiver, a requirement partially fulfilled by the synchronisation process. Secondly, the receiver should be able to recreate the required data for forwarding to the UDSES protocol by the received segments, so it is imperative that a method of selecting the appropriate segments is enforced. The procedure is realised by selecting the appropriate number of messages that can be sent before an acknowledgment is sent back to the sender and at the same time by identifying these segments accordingly. Moreover, since segmentation and multiplexing is available at the TX_LAYER, FCNS provides the means of identifying the datagrams belonging to the same group, or the same transmission connection, and also those

multiplexed onto an EE_LAYER connection. As is described in the TX_LAYER Service Definition section, the TX_LAYER provides this service throughout the data transfer process by using the necessary identifiers in the datagrams.

Finally, the TX_LAYER has the task of imposing the end-to-end security services dictated by the SL. These are agreed via the 10-way handshake mechanism that takes place during the connection establishment phase and aims at providing the necessary means of enhancing the security of the system. The mechanism is described in Chapter 5 in relation to the various layers of the FCNS architecture. By providing security parameters to the TX_LAYER protocol, users may encrypt their data on an end-to-end basis, ensuring the authentication, integrity and confidentiality of the connection throughout the transmission. It is recommended that the end users secure their messages using end-to-end encryption to protect their data against unauthorised disclosure or modification attempts. Interruption and insertion attacks cannot be avoided with this particular mechanism, yet the process offers a great degree of protection against replay attacks, minimising the possibility that an attacker would use any knowledge of the connection to replay part of it or in whole. The TX_LAYER is responsible for initiating the handshake mechanism upon instruction by the SL. It is the layer that will notify the SL of any problems that may arise throughout its deployment, such as corrupt or missing messages. In FCNS, two consecutive unsuccessful handshake attempts result in the connection termination and the parameters securely been reset, destroying the SCs created for the transmission.

The responsibilities though of the TX_LAYER protocol as far as security is concerned do not only involve the end-to-end message security. The TX_LAYER must be able to secure internal messages intended for the UDSES layer as well as the EE_LAYER and the SL, all with different keys to ensure the protection of the layers in case the communication with another FCNS layer is compromised. The level at which such an attack could be launched falls down to the software and hardware

implementation of the FCNS, yet the possibilities that such attempts would be made cannot be negligible.

4.3.3.2 TX_LAYER Service Definition

TX_LAYER service realisation forms the very essence of the TX_LAYER, which promotes the end-to-end capabilities of FCNS to its users. The functions of this protocol as described in the TX_LAYER protocol definition section are used to implement end-to-end session support including:

- QoS management and Monitoring and,
- End-to-end UDSES connection support via:
 - Connection establishment phase
 - Data transfer phase
 - Connection release phase
 - Connection monitoring and management

Connection monitoring and management does not actually form a separate communication stage of the TX_LAYER, since it is an integral part of the transmission. Its differentiation is made only for presentation purposes of the thesis.

4.3.3.2.1 TX_LAYER Connection Establishment

Connection establishment in the TX_LAYER follows the appropriate request made by the UDSES instance. The TX_LAYER user has to inform the layer of the QoS and synchronisation requirements for the transmission, before setting it up. In order for the TX_LAYER to support the requested levels, the user has to choose between the two classes of the TX_LAYER service available.

Class 1 provides the basic end-to-end transmission service of the user data. If this class is chosen then no error detection will be in effect for the connection, in the sense that the checksum validation process as well as the flow control mechanism of the FCNS will not be used.

Class 2 on the other hand offers error detection and recovery, as well as multiplexing functions for the TX_LAYER user. The basic transmission service is enhanced with the capabilities offered by the FCNS in detecting transmission errors and recovering from them. Furthermore, several transmission connections can be multiplexed onto a single EE_LAYER connection, increasing the performance of the network by minimising the processing time and load it would take for the subnetwork to set up and support several connections.

Initiation of the transmission connection establishment phase is signalled by the SL protocol, which implements the 10-way FCNS handshake mechanism. The TX_LAYER actively participates in the process since it is the layer where the end-to-end security algorithms and keys are negotiated between the peers, according to the security context of the connection. The SL issues the *ENCRYPT_ETE* and *DECRYPT_ETE* messages to the TX_LAYER instances, as a step in the handshake mechanism, with the messages being depicted in Figure 4.14.

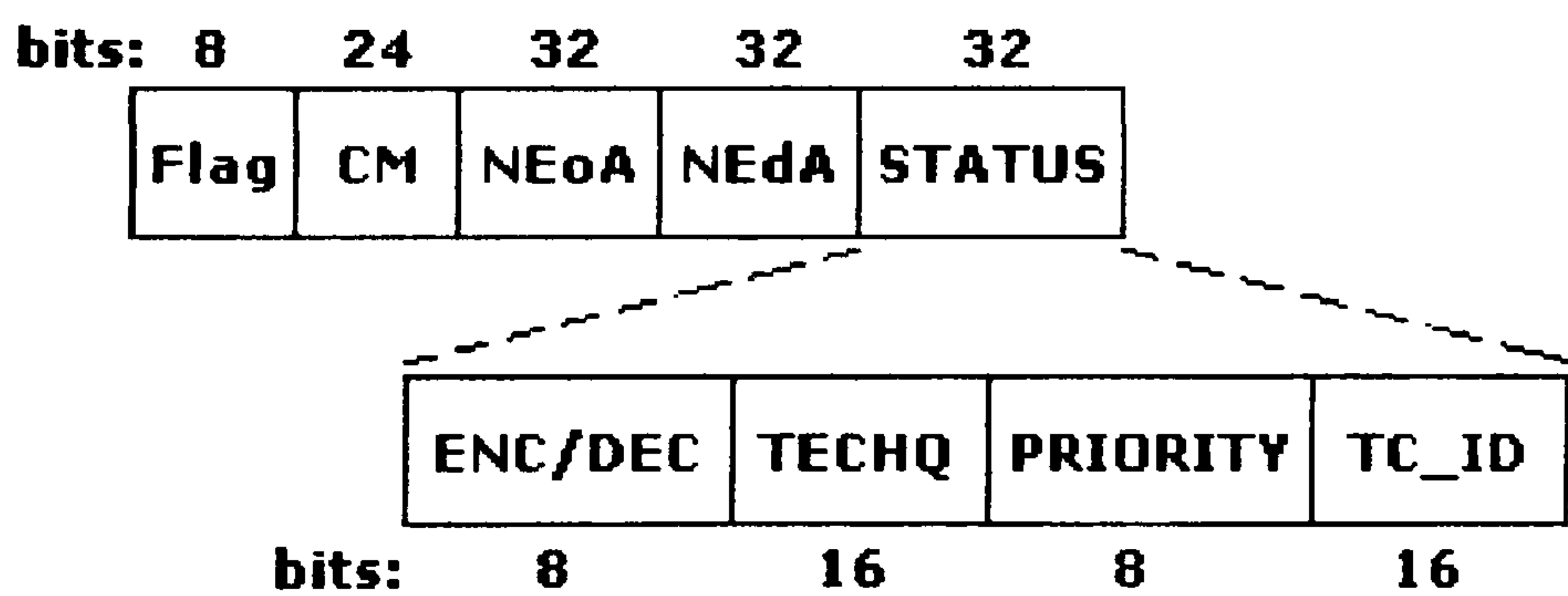


Figure 4.14: ENCRYPT_ETE and DECRYPT_ETE messages

The *ENC/DEC* field indicates whether the message is the *ENCRYPT_ETE* or the *DECRYPT_ETE* one. Information is given as to the algorithm that TX_LAYER will have to use for that connection to secure the datagrams. Secret key information is

transferred in the *TECHQ* field of the message. The field includes facts such as key lengths and whether they form part of a symmetric or asymmetric key encryption scheme. The actual keys are negotiated and transferred to the respective layers during the SC establishment phase of the SL protocol and not with these messages. The *PRIORITY* field contains an indication as to the way the TX_LAYER should process the message immediately upon reception. Finally *TC_ID* field is used to provide TX_LAYER with the necessary connection identifiers that could be used if multiplexing is requested for this layer.

If the handshake mechanism is completed successfully, then the TX_LAYER user will have to choose the service class for the connection. That choice signals the transition of the TX_LAYER from its waiting state to a processing mode, where agreement for the QoS and synchronisation levels for the connection has to take place between the peers. If TX_LAYER cannot support the service required then it notifies the UDSES with the values it can sustain. On the other hand, if the TX_LAYER can support the services in question, then it communicates with the peer the QoS and flow control values that should be used throughout the data transmission.

The values agreed between the TX_LAYER and its user are saved and kept in a temporary memory module, so that they can be cross-referenced with those obtained via the negotiation of the necessary variables with the peer entity. If the receiving entity agrees with the parameters invited by the sender, then it informs the destination of the values agreed, releasing at the same time the memory module from its contents. The TX_LAYER entity then notifies the UDSES protocol that connection can proceed and waits for the data messages to arrive.

Table 4.4 depicts the values that can be negotiated between the communicating entities. These values may change during the data transfer phase as a result of a

situation altering the behaviour and capabilities of a node in supporting the requested QoS.

Table 4.4: TX_LAYER QoS parameters

Parameter	Values
QoS level	Values presented in Table 4.3
Data message length restriction (in bytes)	Values presented in Table 4.3
Timer restriction in acknowledging messages (values are representational)	<ul style="list-style-type: none">- 300 msec- 250 msec- 50 msec- 25 msec
Flow control restrictions	<ul style="list-style-type: none">- 32 message window- 16 message window- 8 message window- 1 message window
Checksum algorithm parameters	<ul style="list-style-type: none">- No checksum (for Class 1 services)- Checksum enabled with method x
Reserved for future use	All zeros

The difference between the *QoS level* parameter values negotiated at the UDSES level (Table 4.3) lays on the fact that transmission connection delay times are greater than the actual time it would take for the message to reach the peer. Users have to deal with issues like acceptance of datagram loss, corruption, duplication and out-of-sequence delivery, prior to the establishment of the connection between the peers. These levels will determine to a great extent the level of QoS that will be maintained throughout the data exchange, although they can be changed in due course if that is requested by a peer.

Additionally, the *flow control restrictions* parameter is used to indicate the number of datagrams that can be sent before an acknowledgement is sent back to the sender. The value can indeed be changed during the data transfer phase, with the TX_LAYER instance storing the outstanding messages on a buffer for their explicit acknowledgement, to avoid unwanted and unnecessary datagram retransmission.

Finally, the *CHECK* parameter represents whether the checksum calculation process will be enabled for the connection and the method that will be used to calculate these values. It is recommended that all TX_LAYER users calculate the datagram checksum prior to its transmission, since it provides a measure in identifying transmission errors that may modify the contents of the data messages.

Upon the success of the mechanism negotiating the QoS and synchronisation parameters for the connection, the TX_LAYER issues a transmission permission to the UDSES protocol, which in turn has to inform the application of the QoS values agreed and forward the data messages to the protocol. The permission is an indication of the TX_LAYER status in supporting the transmission data phase.

4.3.3.2.2 TX_LAYER Data Transfer

The data transfer phase involves the mapping of the data messages arriving from UDSES into the TX_LAYER datagrams and their forwarding to the peer entity, making use of the end-to-end connection service of this layer.

Whenever a MSG_CREATE message arrives, TX_LAYER has to check whether the EE_LAYER can accept the information and the assignment of the TX_LAYER connection into an EE_LAYER one. Supported options in the FCNS architecture include multiplexing, that is, the mapping of many TX_LAYER connections over a single EE_LAYER one, and splitting, that is, the handling of a single transmission connection into various EE_LAYER ones.

If EE_LAYER does not respond within a predetermined period of time, then an unacknowledgement is sent back to the TX_LAYER. Failure in responding to this notice, or in providing TX_LAYER with the necessary parameters to proceed with the data transfer phase will result in an alert sent to the SL and the suspension of the connection, until the EE_LAYER can respond or the communication is released.

Upon success in arranging the above parameters with the subnetwork, the TX_LAYER maps the data passed to it into the appropriate datagram format. The issues surrounding this process involve checksum calculation procedures and the segmenting of the UDSES data, if that is required. The available datagram structure is given in Figure 4.15.

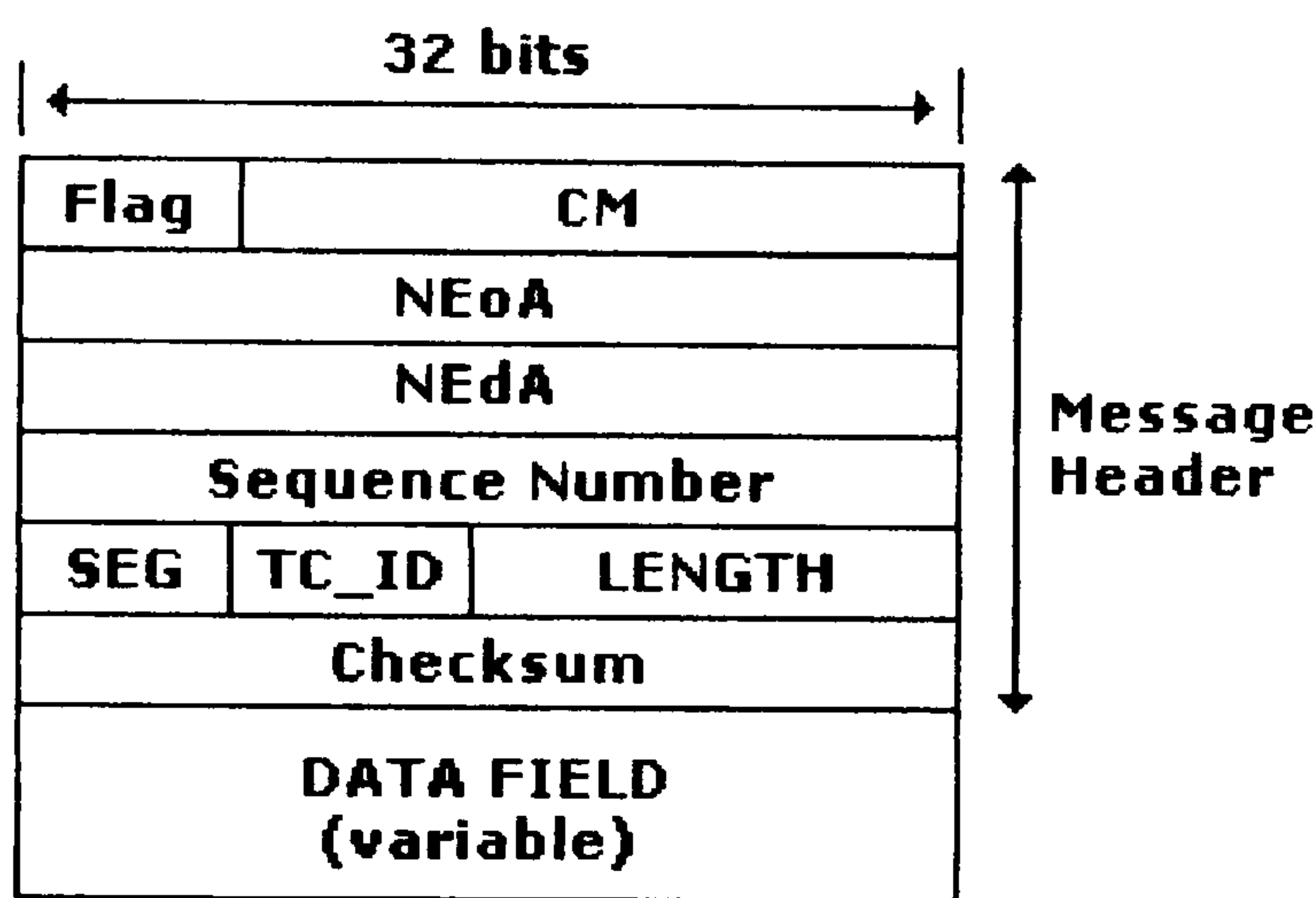


Figure 4.15: DATAGRAM and DATAGRAM_SAR structure

The same message configuration is used for messages transmitted either as segments, forming the DATAGRAM_SAR messages, or in whole, denoted as DATAGRAM messages. Their difference, apart from the length size of the datagram is that segmented messages have the *SEG* field set, whereas its value becomes zero when no segmentation is required. The *TC_ID* field of the message is the value of the transmission connection, which should be used in cases where multiplexing or splitting are used, enabling the receiver to identify the appropriate datagrams for the required connection.

The field denoted *Sequence Number* contains the sequence number of the datagram used to promote the explicit flow control mechanism of the TX_LAYER. Its value is dependent on the number of messages that are selected as the window size and is calculated as: $2^{no.messages} - 1$. The flow control mechanism of the TX_LAYER is similar to the one used in the PHYS layer. Their difference though comes from the complexity of the function from the TX_LAYER point of view, since flow control involves the interaction of the UDSES, TX_LAYER, EE_LAYER and PHYS layer, and also has to compensate for the transmission delay between the TX_LAYER entities, which is usually significantly larger than the actual transmission time. The sequence number though present in the respective DATAGRAM field does not relate to the one used for the PHYS layer frames.

The *LENGTH* field is an indication of the length of the overall DATAGRAM or DATAGRAM_SAR message, so that the peer entities can verify that the restrictions imposed during the connection establishment phase are followed. If the receiver identifies datagrams with sizes larger or smaller than the length agreed then it issues an alert to the SL indicating the error situation. The value of the field is included in the checksum calculation routine, to protect the messages from modifications caused by a faulty transmission link or an adversary, thereby offering an extra protection degree for the FCNS messages.

Finally, the *Checksum* field contains that calculated value, which must be verified at the receiver, prior to the data message forwarding to the UDSES user. The receiving entity resets the value of the field and recalculates the checksum value for that particular segment. If the values match, then the datagram is stripped off its header and is passed onto the UDSES layer. On the contrary, if there is a deviation in these two values, then the message is dropped and an unacknowledgement message is sent back to the sender.

Upon reception of the datagram, receiver has the task of decrypting the message to obtain the actual datagram sent from the peer network element. Decryption takes place using the parameters provided by the SL to the TX_LAYER. If the process is successful then the receiver moves into validating the length size and checksum value of the message, prior to forwarding it to the application. If the decryption process either fails or the obtainable message's header security field is different than the one expected, then an alert is issued to signal the error. Decryption may fail due to a transmission fault altering the contents of the message, or a datagram manipulation with an attacker trying to tamper the security fields. Both cases are handled by the SL, which must notify the system operator of the error situation, which may cause retransmission of the datagram in question and hence create congestion on the network links.

It is worth noticing the effect encryption and decryption processes have on the overall delay of the transmission connection between the peers. As with every FCNS layer, the TX_LAYER sending entity must secure the datagram with the appropriate secret keys and algorithms dictated by the SL. The process has two downsides as summarised below:

- Security headers increase the datagram length
- Security functions add to the datagram processing time

The encryption process produces a message of length greater than the original one, since it includes information concerning its encryption and also random information used for padding, to create a message that would appear as garbage to an attacker trying to intercept it. Additionally, encryption and decryption processes add to the overall time the datagram would take to be processed in the peer nodes, since these functions take some time to complete.

4.3.3.2.3 TX_LAYER Connection Release

Connection termination in TX_LAYER follows the reception of the appropriate request by the UDSES protocol. It can either be an orderly or a disorderly release depending on the situation and the way data transfer has been progressed.

Orderly release follows the acknowledgment of the request by the peer entity, via the NODE_END message. If the receiving entity can release the connection, then it sends a confirmation message back, to indicate that it waits for the SET_STATUS message by the SL to go back to its idle state and reset any outstanding parameters for the specific transmission. The latter may take place in cases where an entity is involved in more than one connection at any time. At the same time, the TX_LAYER informs its service provider that communication can be terminated, so that the EE_LAYER can release any relationship between TX_LAYER and EE_LAYER connections.

However, a TX_LAYER element may not be able to accept the request for the release of the connection. Upon reception of the request it will have to notify the SL of the situation and send an unacknowledgement back to the sender, which will include the necessary justification for which the NODE_END message has been turned down. The reasons governing this action include the pending of outstanding datagrams, which could be delayed by a congested link or a lower rate route than that the NODE_END message was sent through. The sender will have to wait for a predetermined period of time until reissuing the request to the receiving entity. If the receiver is unable to cope with the second request, then the FCNSEP is employed to provide the sender with the appropriate timer values before sending the request again. If the process is unsuccessful after the third try, then the connection is disorderly released and any outstanding datagrams are discarded.

Disorderly release does not form part of the confirmed service provided by the TX_LAYER. At any time, any of the peers may request the release of the

connection no matter the phase communication could be. The release is achieved by issuing the COMM_ABORT message to the peer by means of inserting the message into the normal data flow. Upon reception of such a message, the peer TX_LAYER entity will have to notify the SL of the condition so that it can decide on whether to permit the node issuing the COMM_ABORT message to proceed with clearing the parameters and variables used for the connection from its memory, as well as destroying the SCs set for it. Since the message is related to the release of the end-to-end transmission connection it is recommended that is secured with the end-to-end security algorithm, to minimise the possibility of an attacker issuing the message to force a DoS attack.

4.3.3.2.4 TX_LAYER Connection Monitoring and Management

Connection monitoring and management are actually services offered to the TX_LAYER users throughout all phases of the transmission connection. At any given time any peer may wish to alter the parameters governing the transmission. FCNS offers these services at the end-to-end communication level, via the functions of the TX_LAYER protocol.

The TX_LAYER has the task of preserving the agreed quality of service for the transmission connection, set during the connection establishment phase. This includes the end-to-end transmission of the datagrams at the throughput levels requested by its users, maintaining the transit delay tolerance levels to a minimum. Depending on transmission link capabilities, the TX_LAYER monitors the rates of datagram arrival to the peer entity. The transit and transmission delays and delay jitter are taken into consideration, to respond accordingly in case the acceptable rate falls beneath the agreed threshold.

If the QoS levels must be renegotiated, then the entity that has identified the situation has to issue the appropriate message to its peer. The transmission of the

datagrams should proceed as normal so that data is not lost or further delayed by the process. Upon reception of the message TX_LAYER will have to inform its user of the request made and renegotiate the necessary parameters for the connection. The datagram transmission will be terminated only if the negotiation of the new parameters fails to complete.

Requests for the alteration of the QoS levels for the connection can also be initiated due to an increase of the data messages loss ratio and the TX_LAYER residual error rate. Prior to connection set up, TX_LAYER entities have to calculate an approximation of the transmission failure probability for the connection. If the error tolerance rates agreed do not prohibit the establishment of the transmission connection, then TX_LAYER proceeds in forwarding the connection request to its peer. These rates have to be monitored by the protocol in order to be kept to a possible minimum and hence not to cause problems in the data transfer phase of the connection. If these rates though cannot be maintained, then the connection must be halted so that the nodes can negotiate new parameters for that particular connection. Failure in arranging new error tolerance rates for the connection will result in its release by the TX_LAYER users. On the other hand if an agreement is achieved, then communication proceeds by forwarding the buffered data messages to the receiver.

Management of the transmission connection also implies the monitoring of QoS services such as the end-to-end security of the datagrams. Since the levels of message protection are agreed during the connection establishment process, the TX_LAYER in conjunction with the SL must ensure that those are preserved throughout the communication phases. A user may require, for example, that protection against replay attacks be of no importance for the data messages, instead, passive monitoring and data transfer interruption should be avoided. The protocol has to ensure that the requested services will be supported, by implementing the appropriate mechanisms in the transmission. An example of such

a situation would be the protection of the system against traffic flow analysis. The TX_LAYER will have to work with the underlying subnetwork and implement the FCNS traffic-padding scheme, whereby whenever there are no datagrams to be sent for a certain period of time, the sending entity will have to produce garbage messages that will be encrypted as normal datagrams and sent to the receiver, in an attempt to prevent an adversary from gaining knowledge of the transmission times and patterns.

In all cases, communication will have to be ended if the TX_LAYER protocol becomes unable to support the service levels requested by the UDSES layer. Their renegotiation may not be possible due to strict restrictions imposed by the UDSES protocol, leading to re-establishing the transmission connection via a different subnetwork and EE_LAYER protocol instance.

4.3.4 End-to-End Layer (EE_LAYER)

The FCNS End-to-End layer is responsible for implementing the EE_LAYER protocol, which offers the TX_LAYER datagram transmission services via the realisation of its functions. These can be summarised as follows:

- Reliable link-based transmission of the TX_LAYER datagrams
- Routing and internetworking
- Multiplexing of EE_LAYER connections to PHYS layer ones
- Error detection and recovery
- Error signalling
- Link-based security functions
- Use of PHYS protocol services

The EE_LAYER protocol is responsible for the secure transmission of the TX_LAYER messages to their ultimate destination across any intermediate subnetworks

present in the topology. Its specification diagram is illustrated in Figure A.6 of Appendix A.

4.3.4.1 EE_LAYER Protocol Definition

The task of the EE_LAYER is to ensure that the TX_LAYER messages are appropriately mapped into packets and are assigned to the suitable EE_LAYER connection. Since multiplexing and splitting are options that the EE_LAYER supports, the EE_LAYER must provide the TX_LAYER with the necessary details during connection establishment about supporting the connection and also the required QoS set by the peers.

The reliability of the EE_LAYER transmission phase is dependent on the correct implementation of its functions and on the fact that any restrictions imposed by the protocol are kept throughout the packet exchange process. There may be cases where the underlying physical medium supporting the packet transmission is not reliable, introducing errors such as packet loss and message modification and is also susceptible to passive attacks launched by an adversary. The EE_LAYER must provide the necessary services that minimise the possibility that such errors may occur, but more importantly, that it would recover if such conditions even arise.

Whenever a packet is received by a node, it has to be acknowledged by the receiving entity no matter whether the node intercepting it is the ultimate destination or an intermediate router. This is a measure intended for providing the sender with an early notification of any message loss that could occur, minimising the time that it would take for its retransmission by waiting for its acknowledgment directly from the receiver, which may be several subnetworks away. Additionally, to reduce the load on the network channel that could build up by the insertion of the message acknowledgements, a technique referred to as *piggybacking* is supported in the FCNS architecture. In this method, the acknowledgment is

mapped into another message going back towards the sender, hence reducing the processing time at the intermediate routers that would be required to route additional messages. The receiver will then have to send an unacknowledgement back to the sender if it detects a missing packet, only if after a predetermined period of time the message does not reach its destination. Due to the identifiers present in the EE_LAYER packets, the destination will be able to recognize both the missing segments and the TX_LAYER connection to which they belong, notifying its user about the error to avoid the discontinuity of the connection.

The link-based transmission reliability of the EE_LAYER protocol is also enhanced by the use of error detection and recovery techniques. Early detections of fault situations such as message corruption, message duplication and message insertion may lead to an immediate response by the protocol and consequently its recovery, if that is at all possible. The probability failure of the packet transmission may increase as a result of either a faulty transmission link, an unauthorised party or both. The EE_LAYER has to ensure that it can at least detect such cases, if not be able to prevent them, and devise a plan in conjunction with the SL for recovery.

For the realisation of the error detection and recovery services, the EE_LAYER uses a combination of the error detection and link-based security functions the protocol provides. Whenever a message is to be sent to the peer, the EE_LAYER ensures that a checksum value is calculated on the whole of the message prior to its encryption and network transmission.

Recalculation of the checksum value takes place upon the decryption and authentication of the received message pending its dispatch to the TX_LAYER. If any limitations imposed by the underlying subnetwork, such as length restrictions, have successfully been imposed on the transmitted data, then the checksum value calculated at the peer should match the one present in the received packet, given

that its security has not been compromised. If these values do not match, then EE_LAYER checks the type of error that may have occurred during its transmission.

If the packet is successfully decrypted then the security identifier of the field should remain intact. This means that the error detection process has to check whether the message has been received with either a corrupted flag or length field, before shifting into the possibility that the error has infected the datagram mapped onto the packet. By this method, the EE_LAYER ensures that modification errors caused by either a faulty transmission link or an adversary are successfully detected. As a consequence of the error identification, the protocol will be able to recover from it and notify the peer of the situation, providing an action suggested by the SL for its future prevention.

Since there is always the possibility that the error has been produced by an attacker's attempt to manipulate the connection, the EE_LAYER supports the necessary functions that afford link-based security features for the TX_LAYER. All packets are secured on a link basis, meaning that the messages are protected for their traversing through the various subnetworks that may be present in the topology. By this method, even if an adversary was able to decrypt a packet on a router, the only information that he/she could extract would be the next adjacent node the message would be forwarded to, but not the message contents. The FCNS security architecture is further enhanced at the PHYS layer, where encryption is combined with traffic padding mechanisms affording protection against both passive and active attacks aiming at interrupting or modifying the data flow.

Furthermore, the EE_LAYER is responsible for the internetworking and routing of the packets across different subnetworks or network environments. Routing is based on route calculation functions or mechanisms, which fall into two major categories:

- Datagram routing: A very sophisticated algorithm that enables nodes to route messages individually. It is usually not implemented in network architectures due to the expensive equipment that would be required to support the determination of the requested routes for the individual packets.
- Path – oriented routing: It is the most commonly used technique, in which packets traverse through the same route if they are assigned to the same connection. The downside of the mechanism is that route changes caused by link failures are not quickly realised.

In the FCNS architecture both mechanisms can be supported, yet for the simulation of the FCNS stack, path-oriented routing techniques have been implemented. These techniques have been based on the Open Shortest Path First (OSPF) method [14] using the Dijkstra's algorithm for the calculation of the best route, based on the algorithm's ease of use and simplicity. The criteria used in these calculations have been the link delay of the routes, security weights and the number of hops that could be present, in between the sending and receiving peers. This information is usually made available and exchanged via flooding techniques, in which data is sent to all links of the network measuring the performance of the channels in question in delivering the messages. The parameters are then kept in the routing tables every node possesses and are used whenever a packet is to be directed to another node. These tables can either be static or dynamic, with the latter offering greater performance, since any changes on the links that could affect their performance are automatically being added to the tables. Static tables have the disadvantage that a change in the link delay parameters will result in the downloading of the whole of the table to the nodes, increasing the time it would take to update their respective information.

The EE_LAYER protocol is responsible for the calculation of the routes that exist between the peers and the selection of the most appropriate one for a particular connection. There may be cases where it is important for the application that data

is transferred with the minimum delay possible and situations where errorless transmission is of maximum importance regardless of the delay packets may experience. During the connection establishment phase the EE_LAYER has to ensure that the user requests can be fulfilled and devise a plan for their support in conjunction with the SL protocol.

The process is initiated by implementing Dijkstra's algorithm for the given network topology. The results of the route calculation will be saved in a temporary table of the node commencing the mechanism, which will inform the SL protocol about the route selected for the packet transmission. The SL will verify the validity of the nodes present in the chosen path, if the network can support that process, and reply to the EE_LAYER as to the acceptance or rejection of the route and their respective nodes. If the chosen route is rejected, then the EE_LAYER selects the next alternative from the list for another verification. Failure in obtaining a clearance from the SL will result in the termination of the particular connection establishment phase and the issuing of an alert to the system indicating the error situation. On the contrary, acceptance of the path choice made by the EE_LAYER will result in the information being removed from the temporary table and saved in the respective routing table.

The route calculation process is directly related to the internetworking issues of the FCNS stack, where the intended data recipient may be located on a different network environment to that on which the sender resides. Path identification procedures identify the network to which the destination belongs and enable the EE_LAYER to devise the appropriate route for the connection. If the network environment can be supported, then the SL protocol will forward the suitable address formatting rules to the layer, whereas in any other case it will instruct EE_LAYER to find an alternative network or subnetwork through which messages should be routed. The reasons for rejecting a particular subnetwork or network

environment vary from unidentified nodes to suspicious (in security terms) networks.

An additional EE_LAYER protocol function represents the error signalling procedures of this layer, which are used for the signalling of any conditions that may occur. The process of signalling the TX_LAYER of particular fault conditions is initiated in the presence of either unrecoverable or specific for the TX_LAYER errors, so that the EE_LAYER user can be made aware of the problem and perhaps request the alteration of the communication services offered by that time. In contrast, error signalling between the peer nodes involves the identification of errors including route calculation failures, segmentation/reassembly faults, miscalculated checksums, wrong security identifiers and the like. For both cases, the FCNSEP is implemented as described in Chapter 6.

Unrecoverable errors detected by the EE_LAYER usually involve conditions whose effect is catalytic in increasing the failure probability of a particular EE_LAYER and consequently TX_LAYER connection. Due to the fact that the EE_LAYER instance detecting the error is unable to recover from the condition, it has to signal its user for guidance as to the way transmission should proceed, that is, either the negotiation of a different set of parameters, or the release of the connection. Table 4.5 indicates those faults that may lead to such a condition and their respective meaning inside the NODE_ALERT message issued to the SL.

Table 4.5: EE_LAYER recoverable / signalled errors

Error field	Meaning	Recoverable / Signalled
FATAL	A fatal error	Signalled
ROUTE_CALC_FAIL	Route calculation failed	Recoverable / Signalled
PROT_ERROR	Protocol procedural error	Signalled
DEST_UNREACH	Destination cannot be reached	Recoverable / Signalled

NODE_SUSPECT	Node is suspect	Signalled
DOWN_NODE	Node may be down (hardware)	Signalled
NET_DOWN	Network is down / not available	Recoverable / Signalled
NO_SUBNET	Subnetwork does not exist	Recoverable / Signalled
ERROR_NODE	Error in the node (software / hardware)	Signalled

Signalling of these errors does not imply that EE_LAYER cannot recover from them, and similarly that peer signalling does not exclude their notification to the TX_LAYER. Indication of recoverable errors takes place whenever such an error requires the re-evaluation of the QoS levels requested by the TX_LAYER, since it could directly affect the datagram exchange process. On the contrary, notification of only the EE_LAYER user denotes that the transmission has to be released and the TX_LAYER connection be assigned to another EE_LAYER connection. Usually, recoverable errors are not immediately signalled to the TX_LAYER, since communication can proceed without the user being made aware of the error condition.

Finally it is the responsibility of this protocol to ensure that the services of the PHYS layer are used as dictated by the parameters governing the connection, to facilitate the transmission of the TX_LAYER datagrams. Functions used for this purpose include the assignment of the EE_LAYER connections to their respective PHYS layer ones, as well as the implementation of the traffic padding mechanism of the PHYS layer, which enhances the system's protection against traffic monitoring and analysis.

4.3.4.2 EE_LAYER Service Definition

The EE_LAYER functions are used to realise the services offered by the protocol, enabling and supporting the TX_LAYER communication process across the networks data has to be transferred through.

The services of the EE_LAYER protocol can be summarised as follows:

- QoS parameters support for the TX_LAYER communication
- Error notification
- EE_LAYER communication support
- TX_LAYER internetwork data support
- Security services
- Access to PHYS layer services
- Connection monitoring and management

The services carried out by EE_LAYER protocol are offered to the TX_LAYER throughout the EE_LAYER communication phases, including:

- EE_LAYER connection establishment
- EE_LAYER data transfer
- EE_LAYER connection release

Monitoring and management of the EE_LAYER connection is supported in all communication phases of the protocol, varying from QoS monitoring, routing information updates and error signalling.

4.3.4.2.1 EE_LAYER Connection Establishment

The connection establishment phase is initiated by the transition of the EE_LAYER protocol status from the initial idle to the processing state. This alteration is signalled by the SL protocol in the form of the SC information exchange, related to the EE_LAYER only.

Upon reception of the necessary information that will be used for the appropriate link-based packet encryption/decryption functions, the EE_LAYER confirms their reception and stores information obtained for the peer entity, as dictated by the TX_LAYER. The EE_LAYER has the task of calculating a suitable route towards the destination enabling reliable EE_LAYER transfer of the FCNS datagrams.

The process is initiated by implementing the OSPF method realised by Dijkstra's algorithm. The criteria used to calculate the best available path to the ultimate receiver include a security metric, which should be available to the FCNS stack via the network operator, in addition to the link delay information. The former is used to enhance the selection process, by identifying not only the faulty or heavily congested and delay susceptible links, but also channels that have been classified either as suspicious or unreliable. This categorisation involving link security information can be based upon data exchange process experience and the underlying medium technology supporting the FCNS operation.

Upon identification of the routes that could be followed by the EE_LAYER packets, the protocol signals to the SL the results obtained, which in turn initialises a check sequence to verify the validity of the nodes present in the chosen path, and advise the EE_LAYER accordingly as to the acceptance or the rejection of that selection. In the FCNS architecture, the use of two metrics offers the advantage of a multiple path provision, where several paths could be used to transmit the EE_LAYER packets in case the primary route fails. The variety of the weight calculation enables the EE_LAYER to support the TX_LAYER connection according to the QoS requirements set by the EE_LAYER user.

If the SL protocol can verify and support the connection via the chosen path, then it replies with an acknowledgment to the EE_LAYER. If further information has to be sent as to the nodes present in the route, then that request is included in the respective field of the service primitive message. If for any particular reason the

path chosen is either invalid or cannot be supported by the FCNS architecture, the SL must notify the EE_LAYER, indicating the reasons for which the chosen route has been dismissed.

Success in verifying and accepting the routing options will result in the EE_LAYER being notifying the SL of the format of the packet address fields for transmission. The response of the SL can either be an acceptance of the formatting options indicated or a rejection of the mechanism. The reasons for which rejection of the selected address format procedures may occur, are given in Table 4.6, together with the action taken by the EE_LAYER.

Table 4.6: Reasons for rejecting address-formatting requests

NEI_REJECT value	field	Meaning	Action by EE_LAYER
NO_SUCH_NODE		No such node exists in the topology	Obtain new parameters
NO_SUBNET		No subnetwork present or unknown	Obtain new parameters
NO_SUCH_LINK		Link included on the routing list is unavailable	Route must be re-calculated
DOWN_NODE		Node is not available (hardware fault)	Node is down, signal the SL and the peer entity
NODE_SUSPECT		The node is suspicious / unknown	Node is suspicious, signal the SL and the peer entity
NODE_NOT_CAPABLE		Node is not capable of supporting the connection	Obtain new parameters
ERROR_NODE		A FATAL FCNS procedural error at the node	Cannot proceed release the particular

		communication
--	--	---------------

Depending on the rejection reason, EE_LAYER must provide its TX_LAYER user with another alternative, even if that would mean the re-calculation of the routing information. If the error is unrecoverable in the sense that communication cannot be supported in the given network topology, then the EE_LAYER will have to notify the SL protocol of this situation, which will in turn enforce the FCNSEP to signal the condition to the peer. If the FCNS instance identifying the error resides on an intermediate node/router, then the FCNSEP will be used to notify both peers of the situation.

The routing information sent to the receiving SL protocol entity is followed by the FCNS handshake mechanism, which concludes the connection establishment phase for the EE_LAYER protocol. Upon verification that the connection has successfully been set up, the EE_LAYER goes into a waiting state for that particular connection, pending the TX_LAYER data transfer request, to appropriately allocate its network connection resources to the TX_LAYER connections.

4.3.4.2.2 EE_LAYER Data Transfer

The initiation of the EE_LAYER data transfer phase follows the reception of the appropriate service primitive request by the TX_LAYER. If the EE_LAYER can support the datagram transfer, given any length restrictions that may be imposed by the underlying medium, then it replies with an acknowledgment to the request made.

The reception of the ACK message by the TX_LAYER entity denotes that the datagram exchange can successfully be supported and that EE_LAYER is able to accept the datagrams for the transmission. If for any reason the EE_LAYER cannot accept the datagram, it notifies its user so that the request can be resubmitted.

Table 4.7 provides the parameters used to indicate the reasons for which the EE_LAYER can reject the forwarding of the datagram to it.

Table 4.7: EE_LAYER datagram rejection parameters values

Value	Meaning
FATAL	A fatal error occurred, communication cannot proceed
PROT_ERROR	A protocol error disabling EE_LAYER accepting the connection
SEC_ERROR	Security option not supported
ERROR_NODE	An error occurred in the node (hardware error)

Decryption of the message is done according to the dictations of the SL, which provides the EE_LAYER with the appropriate mechanisms to secure the messages intended for its user. If the process is not successful, then the message is dropped and the request is reissued. Failure in establishing the EE_LAYER status and necessary parameters for the mapping of the datagrams into the appropriate EE_LAYER connection(s) after three consecutive tries, results in the connection's disorderly release.

If the communication can proceed, then the datagram forwarded to the EE_LAYER is assigned to the appropriate EE_LAYER connection, depending on the requests made by the TX_LAYER. The EE_LAYER identifies the next node in the network topology to which message should be routed, and maps the datagram received into the EE_LAYER packet, shown in Figure 4.16.

The *NEoA* and *NEdA* fields of the packet correspond to the network addresses of the nodes that the messages will be routed through, depending on the extracted topology information. The *NC_ID* field is an indication of the EE_LAYER connection

that will handle the transmission of the TX_LAYER datagrams for the specific TX_LAYER connection between the peers.

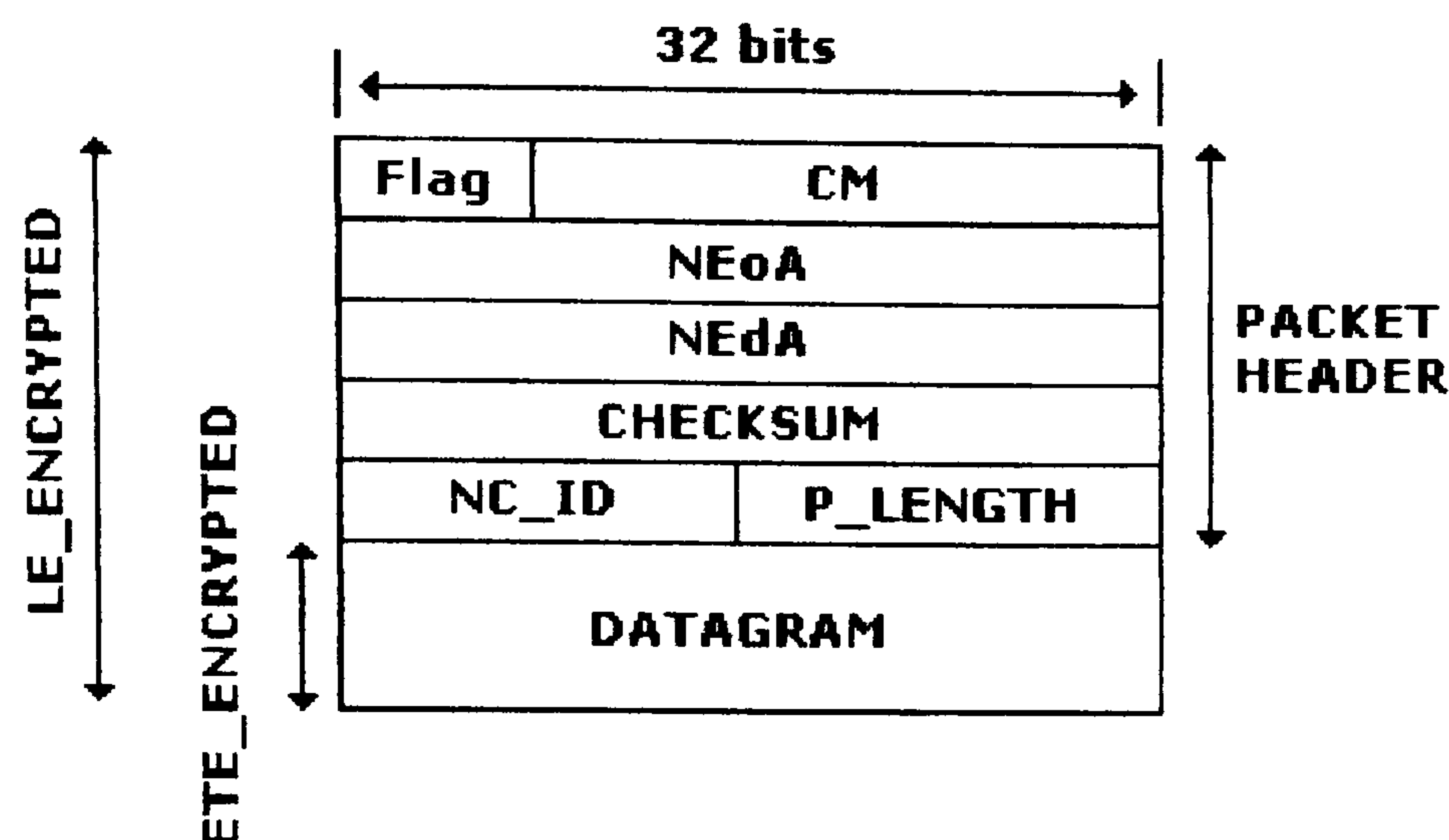


Figure 4.16: PACKET message structure

Additionally, the *P_LENGTH* field holds the value of the packet length, to provide an indication to the receiving entity that any length restrictions have been met, by the sender.

To enable the detection of corrupted packets, a checksum value is calculated at the sender and is included in the packet header inside the *CHECKSUM* field. The value is recalculated at every node where the packet is processed to facilitate the early detection of possible message modifications. If the value received does not match the one calculated, then the message is dropped and the appropriate unacknowledgement is sent back to the sender. If the error persists, then it is signalled to the respective peers via the FCNSEP.

Following the insertion of the checksum value in the packet header, the EE_LAYER notifies the PHYS protocol that the data transfer process is in effect, in order to forward its packets. The message used to arrange the necessary details with the PHYS protocol is realised via the *MSG_FINAL* structure, depicted in Figure 4.17.

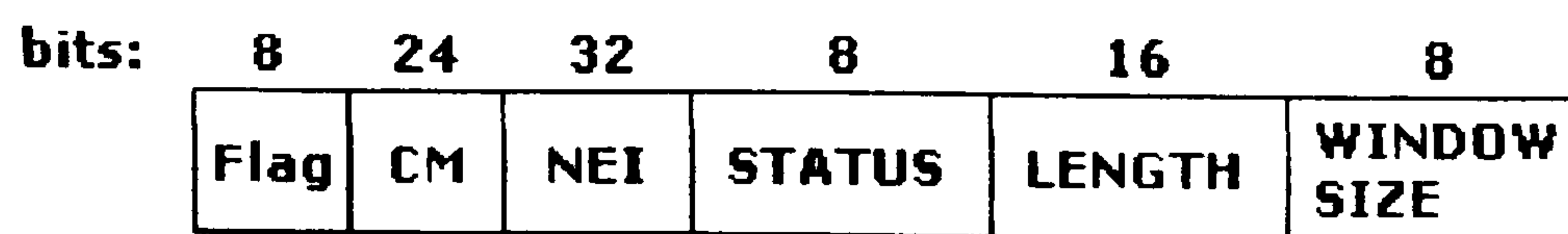


Figure 4.17: MSG_FINAL message structure

The *WINDOW_SIZE* field is used as a measure for enabling the subnetwork to implement the flow control mechanism that will be used throughout the communication. The field indicates the number of messages requested by the application that could be transmitted prior to the receiver acknowledging their reception. As a consequence, PHYS layer will negotiate this value with the EE_LAYER to provide the most appropriate support level possible for the data transfer process. The *STATUS* field contains an indication of the EE_LAYER status as far as the packet format is concerned. Its use is dictated in cases where segmentation may be required due to PHYS layer restrictions on the length size its frames can support.

Mapping of the EE_LAYER packets into the PHYS protocol messages may not be a direct process, in the sense of replacing the frame data payload field with the whole of the received packet. If the application requires the transmission of small sized data, such as signalling information, then several EE_LAYER packets could be placed inside the FCNS frames. The process though may involve the padding of the frames with extra random bits to create a 32-bit multiple PHYS message.

Notification of the parameters' acceptance initiates the forwarding of the EE_LAYER packets into the PHYS layer. The node receiving the packet will have to decrypt its header and verify that the sender indicated in the packet's NEoA field is indeed the one claimed. If the message's integrity has not been compromised by an unauthorised party, the receiving node will check the packet's destination against the routing table it contains in its memory. If the address contained in the NEdA field is not included in that table, then the message is discarded, and a notification is sent back to the sending entity to signal the error. On the other hand, if the

addresses are validated, then the EE_LAYER entity that received the packet will have to verify the checksum value present in the message header.

When the receiving entity is the packet's ultimate destination, then the message is stripped off its header and the datagram is forwarded to the TX_LAYER entity. The mechanism is further supported by the examination of the length field of the received message, where the protocol has to identify whether the sender has obeyed the length restrictions imposed by the underlying medium technology. Failure in obtaining the appropriate field value results in message discard and the signalling of the error to the SL protocol and the peer.

If the data transmission is successful, then the received packet should contain either no errors, or a number of errors that is correctable by the EE_LAYER. Following decryption and authentication, the destination will have to identify the TX_LAYER connections that have been mapped into the particular EE_LAYER connection. The process involves the monitoring of the NC_ID field of the packet obtained, to forward it to the suitable TX_LAYER access point. The EE_LAYER user will have to implicitly identify the TX_LAYER connection to which the datagram belongs and appropriately forward it to the destination instance. Failure to do so after a predetermined period of time shall result in discarding the message, so that the protocol's buffer can be made available for any consecutive packets.

The EE_LAYER enables the monitoring and consequently management of the data transfer phase to maintain the requested QoS level by the TX_LAYER. Communication discontinuity should become an option only in cases where the error is either unrecoverable or when the TX_LAYER is unable to alter the necessary QoS parameters, having received the error signalling by its service provider. The data transfer process is hence orderly terminated when a packet is received bearing the EOF identifier. The EE_LAYER enters a new state where reception of the

NODE_END message should follow, to release the particular network connection and reset the security parameters for the transmission.

4.3.4.2.3 EE_LAYER Connection Release

The NODE_END message indicates that the TX_LAYER end-to-end transmission can be terminated and any associations made between the TX_LAYER connections and the EE_LAYER ones should be finished with the parameters used being reset. Upon verification of the NODE_END request by the SL protocol, the EE_LAYER notifies the peer entity of the connection release demand. The receiving entity will have to check whether there exist any outstanding messages to be received prior to accepting the request. An unacknowledgement is sent back to the sender to indicate pending messages, since the expected packets may have been routed via a slower link than the one the release request has been sent to.

However, if the receiver confirms the release request, the EE_LAYER changes its state to a waiting status, bearing the reception of the SET_STATUS message by the SL. The transition to its initial idle state cannot be completed unless temporary routing tables and the respective path lists for the particular connection are removed from the node's memory. If the layer is involved in another communication process initiated by a different TX_LAYER instance, then it goes back to its processing state to fulfil the requests made by its user. On the other hand, if the layer is able to return to its idle state, it will have to provide the FCNS architecture with the necessary security mechanisms for the link-based encryption of the random packets that may be used, in case the traffic padding mechanism of the stack has been requested.

The EE_LAYER connections can also be disorderly released at any given time by the peer entities. As a consequence, the node identifying a fatal error condition can issue a COMM_ABORT request to the SL instance, which will in turn have the

responsibility of signalling the error to all nodes present on the given topology, via the FCNSEP. By this method, any errors that could affect any future routing calculation procedures, as well as the support of the EE_LAYER data transfer process via the routing of the peer packets, are indicated to the network nodes. Subsequently, the routing tables residing on the nodes can be updated accordingly, minimising the effect such errors may have to data transmissions on that network.

4.3.5 Physical Layer (PHYS)

The FCNS PHYS layer constitutes the interface of the FCNS architecture with the underlying medium where data is to be transferred, offering a reliable and in-sequence data transmission between the network nodes. The protocol is responsible for the transmission of the PHYS layer frames and the management of the PHYS connection, via the realisation of its functions, which can be summarised as follows:

- Synchronisation and sequencing of the data stream
- PHYS communication between two directly linked nodes
- Communication control
- Error detection and correction
- Flow control
- Framing of the EE_LAYER packets for transmission
- Frame routing
- Use of the channel technology for data transmission and connection control

Functions of the PHYS layer are realised by the PHYS protocol, which attempts to provide the EE_LAYER with the services necessary to maintain the required QoS levels for the particular connection, throughout the communication phases. Figure A.7 of Appendix A illustrates the specification diagram of the PHYS layer protocol, for both the sending and receiving layer instances.

4.3.5.1 PHYS Protocol Definition

The PHYS protocol of the FCNS stack architecture has been designed to support the EE_LAYER connections between two directly adjacent nodes, given the current network topology and the route calculated by its user.

The functions used to provide the necessary means for the EE_LAYER packet transmission, include the mapping of the EE_LAYER messages into the PHYS layer frames (framing), PHYS connection establishment, as well as synchronisation and sequencing of the data stream. Error detection and flow control functions are used to enhance the PHYS connection transmission phase, making use of existing knowledge about the underlying channel technology where frames will be in transit.

The PHYS communication establishment indicates the assignment of a particular EE_LAYER connection to the appropriate PHYS one. The PHYS layer should advise the EE_LAYER protocol about the level of support it can provide via its services, given the transmission medium available on a link. If the required QoS cannot be met for the connection, then the PHYS layer must notify its user of the condition and suggest either the arrangement of another EE_LAYER to PHYS connection assignment, or the re-consideration of the QoS levels afforded for the communication.

Synchronisation and sequencing of the network data flow is achieved via the use of control identifiers present in the PHYS layer frames. The messages include fields indicating the exact frame that has been sent, as well as the next one that should be expected. If flow control is implemented, then the frame will include a field identifying the block to which the frame belongs. The receiving entity will then have to verify that the correct messages are received, and wait for any frames arriving out-of-sequence to enable their placement at the respective block. If the receiver identifies a missing segment, then it will signal the error to the sending

entity, requesting the re-transmission of either the particular segment, or the whole block of PHYS frames.

Synchronisation management is carried out throughout the communication process, by means of monitoring the rate at which sender forwards its frames towards the destination, and similarly the rate at which the receiver is able to accept those. To avoid possible re-synchronisation via the FCNSEP protocol that would introduce extra delay on the communication due to the presence of the FCNSEP messages, the PHYS frames contain an indication of the rate at which the messages or the blocks will be transmitted. This value is actually an approximation calculated by identifying the overall length of the frame to be sent and the channel transfer rate, as dictated by the underlying medium. If the receiver cannot accept the value, then a notification is piggybacked onto a frame acknowledgment message, hence enabling the sender adjust its mechanisms to a more appropriate level. The FCNSEP is only implemented in cases where the persistence and severity of the error lead to an unrecoverable situation, where the peer entities are unable to communicate at a mutually accepted transmission rate. A failure in establishing the required synchronisation levels for the connection will result in its release and the consecutive discarding of any outstanding frames.

The maintenance of the PHYS communication is further enhanced by the flow control functions of the PHYS layer. The mechanisms available in the FCNS architecture are based on the sliding window Automatic Repeat Request (ARQ) techniques, including the Selective Repeat and the Go Back N methods [143].

Selective Repeat ARQ is a highly sophisticated mechanism where each frame is individually retransmitted in case of an error, no matter whether it belongs to a frame block or is transmitted as an individual PHYS PDU. The problem with this approach is that expensive hardware may be required to support not only the identification of the frame that has to be sent, but also its correct placement into

the frame sequence at the receiver. For the FCNS simulation, the sliding window Selective Repeat ARQ mechanism has been considered to measure the network performance in cases where every single frame of the sender has to be acknowledged, introducing extra load on the network and processing delay at the node instances.

The Go Back N ARQ technique is the one most commonly used in current access network architectures such as Local Area Networks (LANs). In this scheme, the receiver acknowledges the frames received only after its window has reached the maximum threshold agreed during the connection establishment phase. This means that if for example the window size has been set to be equal to 32 messages, then the frame acknowledgment should be sent after the reception of the block of the 32 frames sent by the peer entity. The disadvantage of this scheme is that if a frame is received in error and hence is discarded, then the sending instance will have to retransmit the whole block of frames prior to continuing with the transmission of the rest of the frame blocks. The receiver will have to maintain a list of the blocks that have arrived to correctly place them onto the appropriate sequence, pending their retransmission. The Go Back N flow control mechanism has also been implemented in the FCNS simulation environment for performance measurement issues.

An additional PHYS connection support is realised via the error detection and correction mechanisms of the PHYS protocol. The technique used in this layer is the Cyclic Redundancy Check Code (CRC) algorithm [144], which is used to protect the PHYS frames from bit pattern modification, resulted either from a faulty transmission link, or an adversary launching a modification attack. In the FCNS implementation the frame checksum calculation process has been based on the CRC-32 algorithm, which has been used to enable the receiver to identify any errors present on the whole of the frame structure.

CRC-32 provides for a low failure probability in the order of 1 in 2^{32} (or 2.3×10^{-10}) chance that a message arrives corrupted at the receiver. Its use has been widely spread due to its ability to detect single bit errors, two-bit errors, errors with an odd number of bits, as well as burst errors that may affect the whole frame [145]. Following the checksum calculation for a particular frame, PHYS protocol appends the value to the message prior to its transmission. The receiver then recalculates the CRC-32 checksum of the received frame and verify the received value with the one computed. If there is a mismatch between these two parameters, then the receiver will check whether the error can be corrected, so that retransmission of the frame in question can be avoided. If the error is unrecoverable, then the message is discarded and the error is signalled to the peer entity.

The CRC-32 checksum computation takes place in the data transfer phase of the PHYS connection and forms part of the connection management capabilities provided by the PHYS protocol. Its choice for the FCNS implementation has been based on the wide acceptance of the mechanism, with additional available techniques, including turbo codes [146] and ADLER-32 calculations [147] also being of interest.

Further to the provision of connection monitoring and management services, the PHYS layer has the responsibility of routing the frames towards their intended destination. The PHYS layer connection involves the transmission of the EE_LAYER packets onto the FCNS frames between two directly adjacent nodes. The EE_LAYER provides the necessary information as to the path chosen for the data transmission, enabling the PHYS layer to appropriately format the address fields of the frame header and afford the requested security services to the outgoing messages.

The PHYS protocol will have to identify the channel capabilities in transferring the requested information, including issues such as the channel capacity and the Round Trip Time (RTT) delay. Their association in calculating the frame sending and

receiving rates is essential, since frame transmission is directly affected by the quality of the communication provided by the underlying medium. At the same time, knowledge of the RTT of the channel provides an indication as to the time it would take for the frame to reach its destination and that of the acknowledgment to come back to the sender. It can be seen that their calculation offers great support in providing the necessary QoS to the EE_LAYER connection, since the PHYS protocol becomes able to identify not only whether the communication can be supported at that time, but also the level at which this can be achieved.

The communication quality of the PHYS protocol data units can be further improved by affording the PHYS security services to the outgoing messages. Although security at this FCNS layer is not imperative, it is recommended that unless explicitly requested by the application, all frames be secured prior to their transmission. Message integrity and access control are the two notions PHYS layer security functions realise, given the appropriate security parameters by the SL protocol. It is the FCNS upper layers responsibility to enforce the authentication and confidentiality functions of the protocol stack, since they are dependent on the particular application instance and should not be viewed by the subnetwork supporting the connection. The reason behind this approach is that the EE_LAYER and the PHYS layer of the FCNS architecture are involved with the routing of the data or control messages across the various subnetworks that may be present in the network topology, and not the processing of the information transferred in those messages.

Message integrity is a function implemented on a host-to-host basis, since the PHYS protocol instance may be shared among various PHYS users and services, minimising the feasibility of realising the security services of the protocol on a single service basis. Such mechanisms are included in the FCNS upper layers, where connection can only be specific for a single application instance, that is, down to TX_LAYER. At the same time, the PHYS layer provides access control

services via restrictions and limitations embedded on the intermediate nodes supporting the connection. If a node is regarded as suspicious or unknown, then the FCNS stack will disable that node's access from the services the protocol provides, ensuring that any unauthorised requests are filtered out.

Achievement of the security services is realised via the encryption functions used to secure the PHYS frames prior to their transmission. The receiving entity will be able to identify from the frame header whether the message has been tampered with and if the connection request has been made by an illicit node instance, discarding any frame falling into this category. In all other cases the message is forwarded down the route, since the PHYS layer can safely assume that the EE_LAYER mechanisms have ensured message's confidentiality.

As a final measure in protecting the system's integrity from passive monitoring and traffic analysis launched at the PHYS layer, the traffic padding mechanism of the FCNS architecture can be implemented. In this case, whenever the node remains at an idle state, bogus data messages are constructed and processed by the FCNS layers as normal data. Security services are afforded to the random data, which will flood the part of the network for which a user has requested protection against channel monitoring attacks. With the random data traversing the network, the attacker will be unable to identify any transmission patterns of a given communication by listening to the particular route. The disclosure of the random data and hence its differentiation from the valuable user data can only be achieved by overcoming all the FCNS security features applied to the message. Although there is a possibility that such a condition may arise, FCNS implementation ensures that this is kept to a minimum, provided that the FCNS user secures its messages according to the specification of the protocol stack and the recommendations made by the network operator.

4.3.5.2 PHYS Service Definition

Service definition of the PHYS FCNS layer encompasses the issues involved with the provision of the communication support services of the PHYS protocol to the EE_LAYER. These services, which are realised by the PHYS layer functions can be summarised as follows:

- QoS support
- Error notification
- Connection monitoring and management
- PHYS layer connection realisation via:
 - PHYS connection establishment phase
 - PHYS data transfer
 - PHYS connection release

The facilities offered to the EE_LAYER protocol aim at providing for the transmission of the user data across the network nodes, supporting a wide range of communication transmission media, such as coaxial cables and optical fibres. The provision of such services is carried out throughout the whole connection set by the PHYS protocol, which has also the task of ensuring a certain degree of protection against passive attacks. The management of the PHYS connection is therefore an internal part of the communication process, with the EE_LAYER instances being able to request a modification or even the recalculation of the QoS parameters set by the nodes, depending on the level of support the subnetwork can offer.

4.3.5.2.1 PHYS Connection Establishment

Connection establishment in the PHYS layer is initiated by the reception of the appropriate encryption/decryption parameters provided by the SL protocol. The PHYS layer entities identify the connection for which information is intended, and provide the SL with the confirmation of the variables received. In case the particular PHYS layer instance is already involved in another communication, then

the protocol has to ensure that it can first support the additional request prior to allocating the necessary resources for the new association. If the request cannot be met, then the condition is signalled to the SL, which will have to wait for a predetermined period of time before reissuing the call.

An error condition may also arise upon reception of the security parameter primitives intended for an already functional connection. The secret keys used for an already established association cannot be changed, in fear of unauthorised attempts to manipulate the SC allocated for the connection by an adversary. If the alteration of that context is imperative, then the connection has to be released in order for the PHYS layer protocol to securely destroy the previous context, discharge its memory from the residing parameters and identify whether it can continue providing support for the particular connection. In any other case, the error is regarded as FATAL, especially when the same request is received more than three consecutive times, leading to the disorderly release of the communication process.

Success in confirming the security parameters for the particular connection indicates that the PHYS protocol can accept the required service primitive requests. These will provide the layer with the necessary information as to the QoS levels that must be supported throughout the association and also the route messages should follow, to apply the appropriate address formatting rules dictated by the EE_LAYER instance.

Upon acceptance of the required parameters information, the PHYS layer will perform the necessary calculations, knowing the channel capacity and the length of the EE_LAYER packet to be sent, to identify the amount of time it should wait until the transmission of the next frame. Failure in identifying the sending and receiving rates of the PHYS entities, as well as initialising the flow control mechanism for the frames transmission, will result in a notification of the error condition to the SL

protocol, pending the recovery from that situation. If the PHYS instance experiencing the error is unable to recover from it, then the connection will be terminated due to the failure in supporting the necessary parameters governing its realisation.

If the connection variables can be correctly set, the protocol may enter the data transfer phase, waiting for the EE_LAYER packets to arrive from its user. The EE_LAYER connection must be assigned to the appropriate PHYS one, depending on the underlying medium and the level of support it can provide to the association. If for any particular reason the EE_LAYER connection must be assigned to another PHYS layer connection, then this is signalled to the SL, which will negotiate the new parameters with both the EE_LAYER and PHYS protocol, before ordering the layers to continue with the data transfer process. The time it would take for this alteration should be kept to a minimum, so that the application user will not comprehend the condition.

4.3.5.2.2 PHYS Data Transfer

Data transfer of the PHYS layer involves the framing of the packets forwarded by the EE_LAYER, the construction of the PHYS header for the frames' realisation, as well as the calculation of the CRC-32 checksum for the message and its protection using the security parameters sent by the SL.

Framing of the EE_LAYER packets is the process involving the mapping of those messages into the PHYS protocol frames for transmission. The EE_LAYER packets are broken up to form the necessary payload information field of the PHYS frame, for which a checksum value will be computed to enable the receiving entity to identify any bit errors that may occur during the frames transmission. The *FRAME* structure of the PHYS protocol is illustrated in Figure 4.18, which also provides the fields of the control information present in the message.

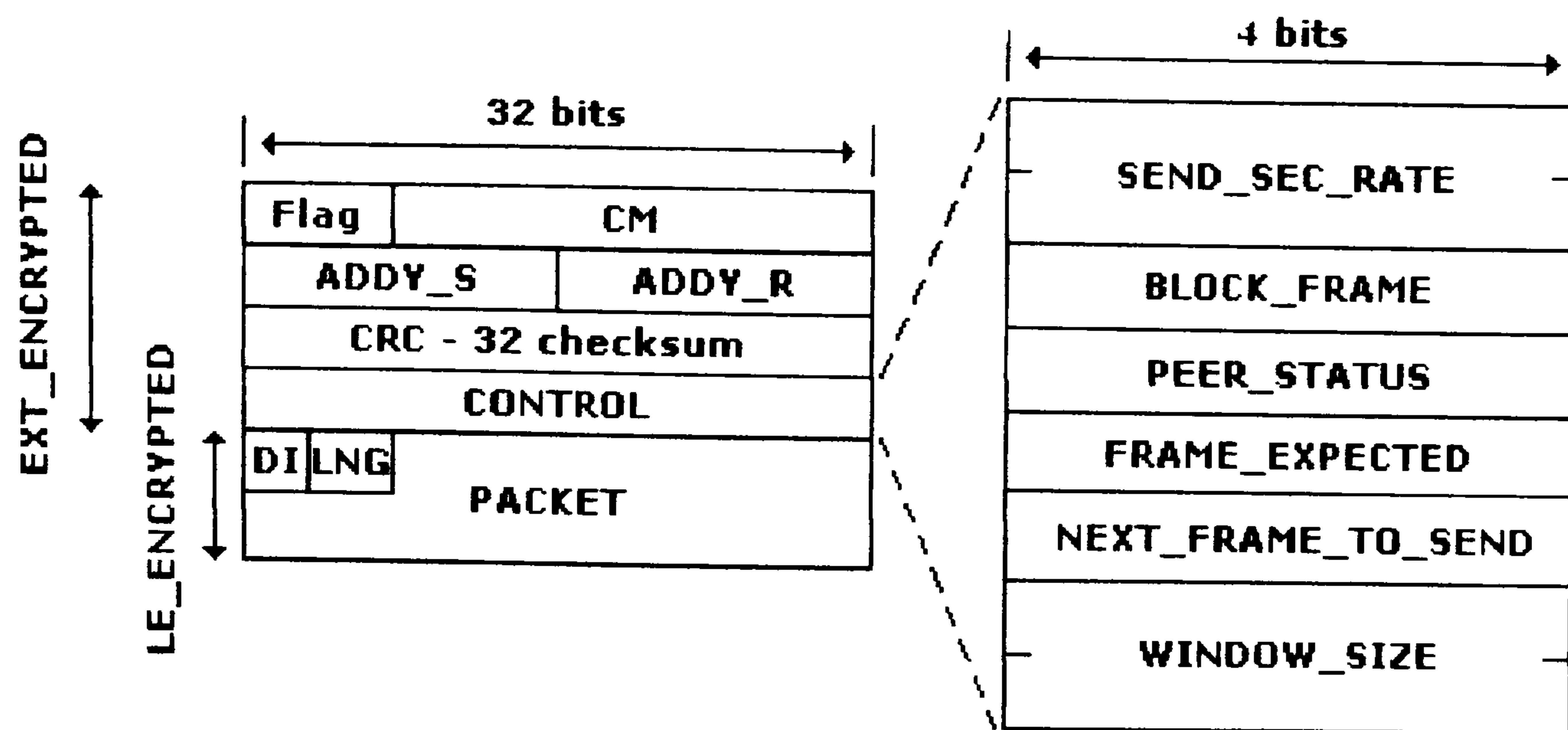


Figure 4.18: PHYS protocol frame structure

The *ADDY_S* and *ADDY_R* fields contain the addresses of the sending and receiving nodes. The frame is directed to inside the network topology, and the route chosen for the particular connection. The *CRC-32 checksum* field holds the value of the CRC-32 checksum calculated on the whole of the frame, whereas its payload area is filled with the link-based encrypted *EE_LAYER* packets.

The *CONTROL* field of the PHYS frame contains various information that are used to regulate the synchronisation and sequencing issues of the data transmission process. The *SEND_SEC_RATE* field holds an indication as to the rate transmitter will send its frames, unless otherwise requested by the receiver. *BLOCK_FRAME* field contains the number of the block this particular frame belongs to, whereas the *PEER_STATUS* field denotes the sender status at the moment of the frame transmission, indicating whether there are consecutive frames receiver should expect, or this one forms the last of the whole segment. The *FRAME_EXPECTED* and the *NEXT_FRAME_TO_SEND* fields provide the identifier of the frame in transit, as well as the one of the next frame receiver should wait for. If the frame sent is the last one for the communication, then this is indicated in the *PEER_STATUS* field, with the *FRAME_EXPECTED* and *NEXT_FRAME_TO_SEND* values being set to 0. The *WINDOW_SIZE* field identifies the amount of messages that should be received prior to acknowledging the frames, facilitating the flow control mechanism of the PHYS protocol. The receiver uses this information to identify the block at which the

frame arrived belongs to, and to discern in case of an unrecoverable error the exact frame that should be retransmitted. Finally, the *DI* field is used to carry the identification of the specific message sent, whereas the *LNG* field represents the length of the overall frame.

Depending on the flow control mechanism enforced by the PHYS protocol, the PHYS entity will either have to acknowledge each frame separately, or after the receiver's window has reached a maximum threshold. The acknowledgment service in the PHYS layer is not imperative, though its use enhances the reliable data transfer services of the protocol to its users.

Acknowledgment of the frame received should come as a result of a series of successful checks the receiver should perform on the particular message. The initial verification comes alongside the decryption of the message and the confirmation that its contents have not been tampered with by an unauthorised party monitoring the connection. Failure in successfully decrypting the frame will result in it being discarded with the notification of the error to the SL, if more than three frames received have been subjected to the same fault.

The unacknowledgement indication may also be sent to the sender as a result of a checksum calculation failure at the receiving entity. Usually, such a bit pattern error is signalled to the sender only if it is unrecoverable. The error notification of both the SL protocol and the peer node should occur in cases where the message is discarded and as a consequence retransmission is required.

Furthermore, the receiving PHYS entity will have to identify the exact segment location which the frame belongs to, to appropriately recreate the EE_LAYER packet for its forwarding to the PHYS user. Following the request for a frame or block retransmission, the PHYS entity will have to maintain a buffer in which the received frames should be placed for sequencing with the remaining data stream. If after a

predetermined period of time the requested frames do not arrive, then the buffer is emptied of its contents for the next block to arrive. This time value should be a function of the RTT delay of the transmission channel, which affects both the frames and acknowledgments transmission. Similarly, the sending PHYS entity maintains a timer used for the acknowledgment of its messages, which should be set according to the RTT value and the window size of the node involved in the association. The window size is the value used to enable receiver to distinguish any frames it has previously received, in case a duplicate is sent as a result of a lost acknowledgment and the resetting of the sender's timer for that message.

In-sequence delivery of the frames will result in the receiver constructing the EE_LAYER packet and its forwarding to the EE_LAYER protocol, waiting for any pending frames that may arrive. If the frame arrived is the last one sent by the peer PHYS entity, then after reconstructing the received packet and acknowledging the frame, the protocol enters a waiting state pending the reception of either the connection release request or another data transfer demand by the particular user. If the layer instance is involved in another communication, then it is only the particular connection for which the connection release request has been issued that should be terminated, enabling the PHYS layer to support any other associations with a peer.

The management services of the PHYS protocol are available at this communication phase, including the connection monitoring, error notification and QoS support of the association established. The services are realised via the PHYS frame, as far as the data transfer process is concerned. The CONTROL field of the PHYS protocol data unit contains information concerning frame delivery, the flow control application by the protocol, as well as the rate at which the sender transmits its packets. Whenever the renegotiation of such parameters is imperative, the new values are denoted on the frame headers sent between the peer entities at any time, on account of the full duplex nature of the PHYS connection.

Embedding of the new parameters requested for the particular connection into the frame header minimises the need for the additional messages to realise the process and the network load that could be increased by the redundant messages. An additional advantage of this reasoning is that connection is maintained throughout the communication of the new parameters and hence the protocol can continuously support the service requested by the application user. Service discontinuity should come only as a result of unsuccessful attempts for the renegotiation of the necessary parameters used to realise the PHYS connection, leading to possible unrecoverable errors for the association. The FCNSEP should be used prior to issuing a disorderly release request, so that the peer is notified of the error situation to update its information about the nodes and the links present in the network topology.

4.3.5.2.3 PHYS Connection Release

Orderly connection release follows the appropriate request made by the PHYS user upon conclusion of the data transfer process. The PHYS protocol will have to reset the parameters set for the particular connection and move into releasing the connection resources used for the communication support. At the same time the entity should check whether it is involved in a communication with another user, so that the PHYS entity does not return to its initial idle state that would result in the immediate breaking down of any other association already set up.

The request made by the EE_LAYER for the release of the PHYS connection is signalled via the NODE_END message, which is also sent to the peer entity for confirming the demand. If the receiving entity can acknowledge the release call sent by the source node, then a confirmation is sent back to the sending instance, whilst the request is forwarded to the SL protocol. The tearing down of the communication is signalled to both peers via the SET_STATUS message of the FCNS SL protocol, indicating that any SCs set up for the connection in question should be

destroyed. If the traffic padding mechanism of the protocol is enforced, then the security contexts are kept for securing the random data messages used for enhancing the system's protection against traffic analysis attacks.

On the other hand, disorderly release of the PHYS connection has to be verified via the SL protocol, for fear of a DoS attack by an adversary. If the verification process is not successful, then the communication party issuing the demand is instructed to continue with the frames data transfer. However, verification of the COMM_ABORT request, signals the release of the association and the transition of the layer to its idle state, given that there are no other connections supported at that time. Again, if traffic padding is supported for the particular application instance, then the peer instances should create random data for transmission across the network topology. In any other case, the PHYS layer protocol will move its state to an idle condition, pending any connection requests by the EE_LAYER protocol.

Chapter 5: Future Core Networks System (FCNS) – Security Layer Protocol

Chapter 5. Future Core Networks System (FCNS) – Security Layer Protocol

5.1 Overview

5.2 FCNS Security Layer (SL)

5.2.1 SL Protocol Definition

5.2.2 SL Service Definition

5.3 FCNS Keystream Generator

5.3.1 Description

5.3.2 Statistical Tests

5.3.3 Cryptographic Tests

5.3.4 Applicability

Chapter 5 provides information relating to the security considerations of the FCNS architecture and the functions used to provide the necessary services to secure the FCNS stack. A general overview of the FCNS security architecture is given followed by an analysis of the Security Layer (SL), depicting its protocol and service definition with respect to the FCNS communication layers. Finally, the FCNS keystream generator is presented, which forms part of the SL protocol, producing the secret keys used for the protection of the FCNS interlayer messages.

5.1 Overview

Network system architectures have evolved greatly since their first establishment in the 1980s, leveraging the design and development of communication protocols used to support their operation. Chapter 3 has described the need for security functions and services and has made reference to the reasons why the protection of the user and/or signalling data has been imperative, leading to a new era of communication rules such as IPv6 [67], ATM [15] and SCTP [66].

The main implication of currently used secure communication protocol structures is the protection of the user data across the Internet and the networks that might route the messages towards their intended destination. In FCNS, the need to enhance these sets of communication rules has been identified, providing a framework where security would be used to safeguard the messages exchanged between the layers of the protocol stack and then to offer the necessary services to enhance the user data transfer process.

The FCNS communication services, presented in Chapter 4, are protected by the security functions afforded by the protocol stack, via the use of the SL protocol. FCNS differs from existing architectures in that message protection mechanisms are offered in all layers of the communication, and are managed and maintained by a single protocol.

The FCNS security architecture was developed to provide a simple design and the ability to modify the security features of the stack without affecting the FCNS communication layers. Furthermore, the mechanism ensures that compatibility with existing architectures would be maintained in cases where an existing set of communication rules replaces one of the FCNS layers. An example of the latter case could be the use of IPv6 as the EE_LAYER of the FCNS, or the SCTP replacing the TX_LAYER of this structure.

The SL protocol has consequently been developed to provide the following features and requirements to the FCNS communication process:

- *Confidentiality*, ensuring the connection and data protection against unauthorised access and disclosure,
- *Integrity* for protecting the user and signalling data from either accidental or malicious modifications
- *Availability* to guarantee the reliable and legitimate network resource usage and
- *Accountability* to associate the events and service usage to the specific peers responsible for these actions. *Authentication* and *Non-repudiation* are functions associated with accountability features since they provide for the verification of the users originating and receiving the messages.

In the following sections, the SL protocol and service mechanisms are described followed by the FCNS keystream generator, which provides the necessary parameters for the security context exchange between the SL and the FCNS communication layers.

5.2 FCNS Security Layer (SL)

The realisation of the SL functionality is based on the SL protocol and the tasks the layer accomplishes, as summarised below:

- Security functions including:
 - Peer entity authentication
 - Connection integrity
 - Connection confidentiality
 - Non-repudiation
 - Access control
 - FCNS 10-way handshake mechanism
- Security Context (SC) exchange

- Error signalling
- Interface to the system operator

The following subsection analyses the functions of the SL protocol, whereas the SL Service Definition indicates the services of the SL supported throughout the communication phases. Figure A.8 of Appendix A depicts the abstracted specification diagram of the SL to which its implementation should conform.

5.2.1 SL Protocol Definition

The Security Layer of the FCNS protocol stack affords the required security services to both the internal and external FCNS messages. Its functionality is based on the exchange of the appropriate SCs for a particular connection both for the peer messages, as well as the primitives exchanged between the FCNS protocols.

The application and use of the security functions of the SL is not mandatory at any level of the communication, though it is recommended that the transmission be secured at the highest possible degree. The initiation of the protocol mechanisms can either be triggered by the particular user, or be offered on a basis dictated by the network operator.

5.2.1.1 Peer Entity Authentication

Authentication is one of the most important functions the SL protocol provides FCNS with. It ensures that only a legitimate user can access the services offered by the FCNS stack and the network, denying their admission to any unauthorised parties. For the user data transfer process, authentication implies that only a valid user can send information down the communication channel, which in turn can be obtained and viewed by any listening party.

The UDPRES and UDSES protocols are involved with the negotiation of the information syntax and session synchronisation and management functions of the FCNS connection and hence need not be afforded such services by the SL protocol. This does not imply that any messages originated from these two layers intended for their peer instance or for another FCNS layer will not be secured prior to their transmission. Indeed, at the UDPRES level, data manipulation functions such as compression and encryption may be negotiated between the nodes, as parameters of the context exchange process, taking place during the connection establishment phase. Additionally, security functions apply to the UDSES protocol as well, for the protection of the synchronisation information and the setting up of the secure session between the peers. An example of such function is the Secure Socket Layer (SSL) protocol [148]. The authenticated operation of the UDPRES and UDSES protocols relies on the TX_LAYER and EE_LAYER of the FCNS architecture.

The secret keys used for the EE_LAYER and TX_LAYER instances are negotiated between the peer entities during the SC establishment phase and more specifically the 10-way handshake mechanism. The SL provides the essential information to the protocols, which then negotiate the parameters that will be used throughout the connection, via the use of a session key separate for each layer. The process is usually dependent on either a trusted third party that would provide the appropriate keys for the secure SC exchange, or the network operator supporting this process. Key management and exchange techniques are widely available in communication systems [109], [149], with their design being outside the scope of this thesis. For this work, the existence of an adequate mechanism for the key exchange process has been assumed. The mechanism could be implemented in FCNS if needed, if a key administration authority is offered by the operator for the specific network domain. Release 6 of the UMTS CN supports such a method in the Network Domain Security (NDS) Authentication Framework (NDS/AF) [150].

Authentication functions are also supported on a connectionless service basis, in the form of data origin authentication. In this case, the source and data recipient authenticate only the origin of the data messages, due to the lack of a distinct end-to-end based communication. The notion of a peer entity does not apply in connectionless connections where the operation of the UDSES and TX_LAYER protocols should normally be of minimal impact. This results in the lack of a session connection establishment and the transmission of the data stream via the EE_LAYER connections towards their destination.

The functions supporting the authentication services either for the peer-entity or data origin case should normally be similar, if not the same, in that the algorithms and secret keys used should be identical unless explicitly requested by the network operator. As long as the SCs for subsequent connections are independent of each other, the chances that prior connection knowledge be used to attack another association are dramatically minimised.

5.2.1.2 Connection Integrity

In addition to authenticating the user, it is very important to verify that the information received has not been tampered with. It is of minimal benefit to secure a message and not to guarantee its authenticity and similarly validate its authenticity without verifying the integrity of the message. Figure 5.1 below indicates by means of sample network attacks the reason for which authentication and integrity are issues strongly coupled together.

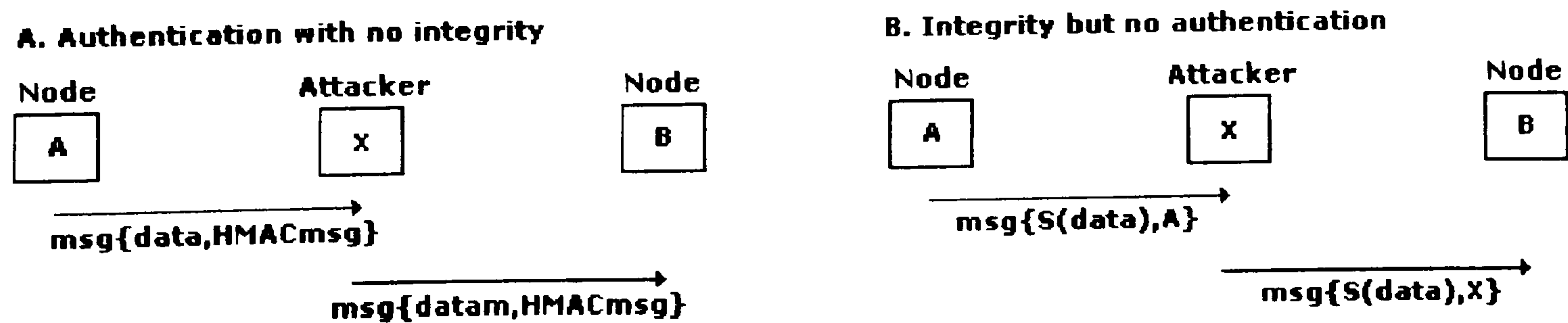


Figure 5.1: Sample communication attacks

In the first case, the peer node A sends an authenticated message to B, via a channel where the attacker X is assumed to have channel monitoring capabilities, as well as the ability to launch an active attack on the user data. Since the message is authenticated, the *HMACmsg* value depicted in Figure 5.1 cannot be altered without the modification becoming unidentified by the peer entity B. The attacker simply alters the contents of the message, from *data* to *datam*, which will consequently reach the recipient in error. Node B will verify that the message actually was originated by Node A, though it may have to discard it due to its tailored contents.

On the other hand, if the message is sent encrypted over the communications channel but lacking authentication measures, then the attacker can intercept it and present the data as its own. No matter what the situation, imposing both authentication and integrity security functions, protects the user data messages in the sense that only the legitimate user can actually forward the information onto the communication link.

Data integrity is available to all FCNS layers, since message security between peers is believed to be of utmost importance for the protection of the interlayer FCNS messages and consequently the user data. Service primitives and requests exchanged between the FCNS protocols and the peers should always be secured, to prevent an adversary from manipulating a connection. In the FCNS architecture it is recommended that integrity functions be employed to verify not only individual messages but also their respective sequence as a whole.

Nevertheless, authentication and integrity functions only ensure the legitimacy of the sending instance among the peers, since any listening party could read the data messages of the channel. To provide the necessary service whereby only a legitimate receiver would be able to interpret the data exchanged, confidentiality functions should be enforced for the particular connection.

5.2.1.3 Connection Confidentiality

Connection confidentiality functions are usually employed in cases where sensitive data is transferred in a network, enabling only the two communicating peer parties to comprehend the messages exchanged. Figure 5.2 illustrates, by means of sample network attacks, why confidentiality is necessary in a network topology consisting of various nodes.

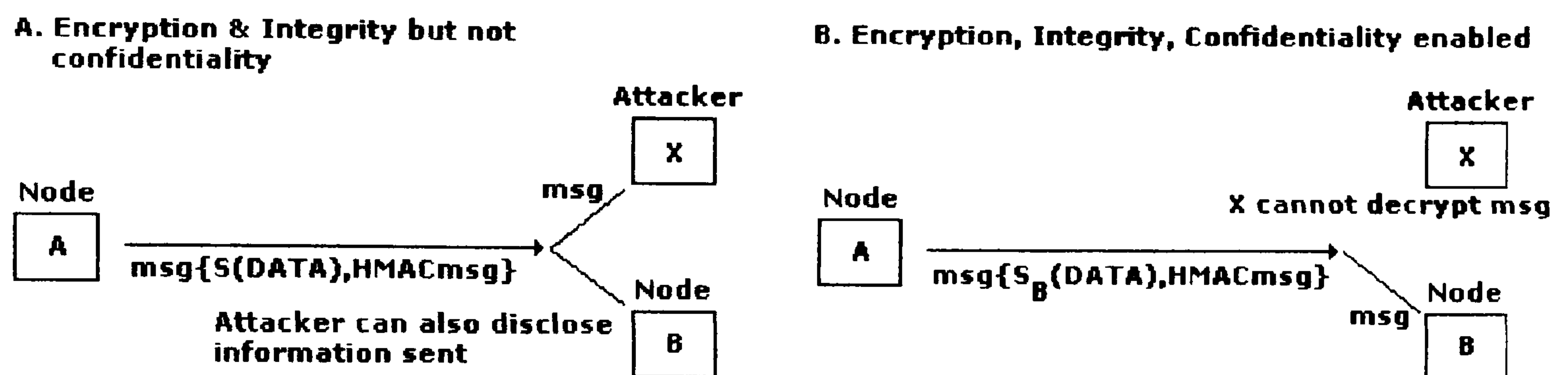


Figure 5.2: Sample attack against message confidentiality

In the first case, the user data is assumed to have been afforded only the authentication and integrity functions of the SL protocol. If the attacker X launches an interception attack, then the messages will never reach the node B, hence succeeding in launching a network DoS attack.

When the messages are only encrypted with a symmetric key algorithm, any party gaining access to that key will be able to decrypt the data and obtain the information transferred to it, breaching the data stream's integrity. The node X can either cause the discontinuity of the connection, or re-insert the packet into the link for transmission to the peer node B. If no alteration is performed on the data payload, the receiver will never be aware that the attack has occurred, assuming a protected communication.

In contrast, employing confidentiality functions in the data transfer process ensures that the messages in transit cannot be manipulated. The peer node A encrypts the packet with an asymmetric key encryption mechanism, whereby only the node B can decrypt the received information and hence prevent any disclosure of sensitive

data to unauthorised parties. The attacker X will not be able to decrypt the messages sent and hence is only able to discontinue the connection, given channel monitoring capabilities and interception privileges. In any case, only the peer node B will understand the enciphered message, thus verifying the confidentiality of the data sent.

In FCNS, confidentiality functions are available to all layers of the protocol stack, providing that integrity mechanisms are used. This is because the same mechanisms can be used to enforce these two services, as their secret keys are completely different. It is recommended that confidentiality is enforced in all message sequences and not only to individual FCNS packets. The latter may come into effect in a connectionless-based transmission, where the SL will provide for data confidentiality services.

5.2.1.4 Non-repudiation

Non-repudiation is an optional function provided by the SL protocol, whereby proof of origin or data delivery is employed prior to the initiation of the connection data transfer process. The purposes of this function can be summarised as follows:

- To protect the sender against false denial of data receipt by the peer node
- To protect the receiver against false denial of data transmission from the sender

It is clear that success in employing the non-repudiation security function is dependent on the presence of a trusted third party acting as an impartial judge for resolving any disputes that may arise. If the network environment can provide such an authority, then non-repudiation can be applied in the application layer running on top of the FCNS architecture. Although it could be used in conjunction with the TX_LAYER protocol, non-repudiation requires the negotiation of certain issues prior to its employment. For example, to protect the sender from false

reception denial, the peer entity should first send a digest or a part of the data that is to be transferred to the receiver, which should in turn reply upon the acceptance or rejection of the connection. The communication should be monitored by a certification authority, which would then provide the necessary mechanism enabling the application of a digital signature to the user data.

Non-repudiation functions are recommended in cases where the sending instance is unaware of the network through which messages travel. The home environment should ensure that if intermediate networks are unknown or suspicious, then the appropriate measures should be enforced to provide protection from false service denial by an illicit node.

5.2.1.5 Access Control

The final security function category supplied by the mechanisms of the SL protocol is the access control operation, in which the network administrator wishes to identify and check the user privileges in setting up and accessing the various services offered. The issue is usually invoked alongside network management procedures, since at any time the provider may request the examination of the users accessing the network resources and/or services.

Access control implementation is dependent on the authentication mechanisms for the user or network node instances, since authentication is the measure ensuring that only legitimate subscribers can actively request access to network services, including data transmission. Nevertheless, the two notions should be separated since the legitimacy of a user does not necessarily imply permission to access all services.

In the FCNS communication architecture access control functions are available to a number of layers, from the PHYS layer up to the application running on top of the

stack with realisation dependent on the available hardware in the network. A possible scenario would be the implementation of such functions in network routers limiting the network addresses for which service access is unconstrained, in addition to filtering at the intermediate bridges of an Ethernet network supporting the communication.

Since access control is a strongly application dependent function, it has not been considered in depth in the SL protocol implementation. Although support is provided for such services, their provision does not imply any necessity in a peer environment routing the messages towards their intended destination.

The appropriate secret keys and algorithms required to support the SL protocol functions are distributed to the FCNS layers during the SC exchange procedure at the connection establishment phase. The context contains all the information relevant to the particular layer or the peer to which information is sent. Communication should not proceed without the transfer of the necessary data that will be used to afford the security services of the SL, since such an action would imply the vulnerability of the connection against both passive and active protocol and network attacks.

The SC exchange is realised by the provision of the 10-way handshake mechanism initiated by the SL protocol, given in Figure 5.3. The FCNS communication layers provide the connection request for the connection establishment to the SL, which identifies whether the same request has already been made for the particular connection. If such a demand has been made twice then the communication setup phase is released as an error has occurred. Otherwise, upon successful verification of the request, the SL initiates the SC exchange between the FCNS layers and the peer entity.

Security contexts are created for each FCNS communication layer separately, with the keys involved being different and independent of each other. The SC associated with the peer-to-peer connection protection is a distinct context, which should not be correlated with any other parameter set for the connection. If the system becomes unable to produce the SC for the end-to-end communication due to a possible protocol procedural error or a software fault, then depending on the necessity of the particular connection, this will either be terminated or user/signalling data will be sent in cleartext format. It is recommended that such a connection should never be established to avoid interception and modification attacks on the data in transit.

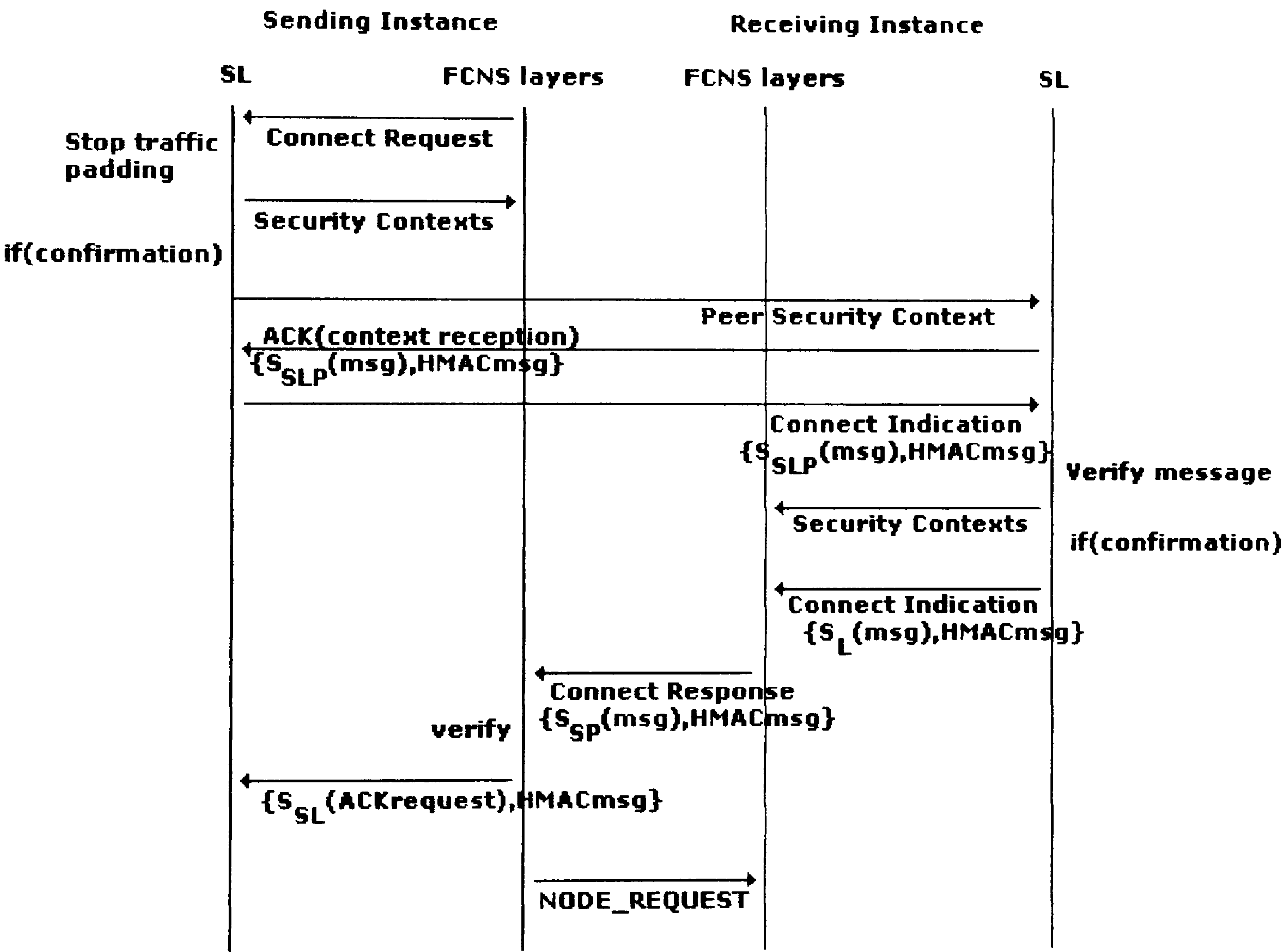


Figure 5.3: FCNS 10-way handshake mechanism

Table 5.1 depicts the notions used in Figure 5.3 together with the explanation of the messages exchanged. Those messages should contain identifiers specifically assigned for the connection, to offer the system protection against possible replay attacks.

Table 5.1: Message explanation for the FCNS 10-way handshake function

Messages	Meaning
$\{S_{SLP}(msg), HMACmsg\}$	Message secured (integrity and confidentiality) with secret key specific for the peer SL, message authenticated
$\{S_L(msg), HMACmsg\}$	Message secured (integrity and confidentiality) with secret key specific for the FCNS layer, message authenticated
$\{S_{SP}(msg), HMACmsg\}$	Message secured (integrity and confidentiality) with secret key specific for the FCNS peer, message authenticated (in this case for the sending instance)
$\{S_{SL}(msg), HMACmsg\}$	Message secured (integrity and confidentiality) with secret key specific for the SL, message authenticated

Following the completion of the SC negotiation, the SL instructs the entity to proceed with the NODE_REQUEST message transfer, concluding the FCNS handshake mechanism and signalling the beginning of the connection establishment process. Failure in completing any of the verification and confirmation functions of the FCNS stack during the handshake procedure will result in the release of the connection and the re-issuing of the connect request by the node. This action implies the destruction of any old SCs and the negotiation of a new set of parameters for the connection.

The security functionality of the SL protocol is ended during the connection release phase of the communication, where all SCs associated with that connection are destroyed and the parameters used erased from the node's memory. Failure in completing this phase will result in an alert being sent to the system operator, since used keys should be removed from the system so that an attacker cannot use them to force a replay attack.

The SL protocol is also involved with the mechanisms ensuring the secured signalling of any errors occurring throughout the communication phases, via the implementation of the FCNSEP. Error signalling procedures are triggered upon reception of a NODE_ALERT message issued by any of the FCNS communication layers, as described in Chapter 4. The message may be the result of a protocol procedural error, whereby a message sent by another layer may be received in error, or a condition that has occurred at the peer, initialising the FCNSEP to inform the entity of the fault situation. The SL protocol has the task of intercepting such messages and identifying the error so that its correction may proceed. The SL should identify the fault condition and attempt to devise the appropriate corrective action, without notifying the system operator about the circumstances that initiated the transmission of the FCNSEP or NODE_ALERT messages.

Usually, the network provider should supply the SL protocol with a list of recommended actions associated with a particular error condition, such as for example the recalculation of a given route in cases of a subnetwork failure. Only errors falling into the fatal category should be signalled to the system, following the release of the connection for fear of an active attack by an adversary.

As a final remark it should be noted that the SL protocol has been designed to serve as the interface of the FCNS communication layers with the system on which FCNS is implemented. Security and network management functions are uploaded to the node's memory and are consequently handled by the SL protocol, which will decide on the FCNS layers' functionality whenever the particular instance is actively participating in a connection.

5.2.2 SL Service Definition

The functions of the SL protocol are realised throughout the communication phases of the connection, implementing the services of the protocol for use and access by the FCNS communication layers and the peer entities. These services can be summarised as follows:

- Security services such as authentication, confidentiality, integrity, non-repudiation and access control
- Security services to the FCNS communication layers
- Traffic padding services
- Connection monitoring and management via the
 - SL connection establishment
 - SL data transfer monitoring
 - SL connection release phases

The connection monitoring and management service is in a sense the most important one offered by the SL protocol, since it involves the applicability and enforcement of the SL security functions for the particular connection. The SL service definition is presented in the form of the communication phases of an FCNS instance in relation to the FCNS communication layers and the requests intended for the peer entities.

5.2.2.1 SL Connection Establishment

FCNS stack functionality is subject to connection requests made by an application entity, either on behalf of a subscriber or a particular network element transferring signalling information to a peer network node. Assuming that the protocol resides in its idle state, then the necessary resources can become available to the node issuing that request, with the SL protocol being able to support the consequent connection and offer the required services to its users. In contrast to the FCNS communication layers where a protocol user is the layer protocol directly above it,

the SL constitutes the service provider of the whole FCNS stack structure, arranging security issues for each layer independently.

Upon reception of the connection request, the SL initiates the handshake mechanism for the SC exchange between the FCNS layers and the peer entity. If the definition of an SC appropriate for the connection cannot be made, then the request is dropped and the protocol returns to the idle mode. Otherwise, the SL initiates the FCNS keystream generator, in case original secret keys do not exist for the SCs to obtain the necessary parameters used for the protection of the FCNS interlayer messages. Simultaneously, the protocol identifies the presence of any Certification Authority (CA) in the network, which will provide the keys for the SC between the peer nodes.

Success in generating the appropriate contexts for the connection results in their exchange and confirmation by the instance's communication layers and the intended data recipient. If the context can be accepted, then the connection request is forwarded to the destination, which will decide upon its acceptance or rejection. The SL protocol will have the task of monitoring the exchange of the service primitives required to complete the communication setup process and provide the FCNS layers with the appropriate security services to fulfil the user requests.

As far as the application instance is concerned, the SL protocol has to identify that the data transferred is afforded the necessary security facilities, such as authentication, message integrity and confidentiality. If non-repudiation provision needs to be made, the SL will have to negotiate the service with the CA and send a data digest to the recipient, which will in turn inform the SL protocol and the authority upon the acceptance of the information that is to be sent. The realisation of the SL security functions to the application instance is subject to the correct enforcement of the SC exchanged between these two entities, with the SL protocol

identifying that the service primitive to be sent has indeed been provided with the requirements set by the user. This implies the interception of such a message prior to its forwarding to the receiver in an attempt to identify any miscalculated security headers, false encryption, inadequate authentication and the like. If the message successfully passes the check routine, an acknowledgment is sent back to the instance, which then forwards the message to the FCNS communication layers for transmission. Persistent failure of the application in correctly enforcing the security rules imposed results in the connection's release.

For the UDPRES connection establishment phase, the SL protocol assists the UDPRES layer in protecting its messages against possible attacks by using the parameters obtained by the SC for issues including message integrity and possibly confidentiality. Usually, encryption issues in the UDPRES layer are negotiated with the peer during the context exchange process, where ASN.1 encryption extensions may be used to facilitate the enforcement of the network's security policies to the connection [151]. Nevertheless, UDPRES messages should always be secured for integrity and confidentiality reasons using the appropriate keys provided by the SL. Confidentiality may not be requested for this protocol, since information regarding UDPRES is only visible to the ultimate destination and not any intermediate network nodes.

Assisting the UDSES layer in establishing the session between the peers and negotiating the appropriate QoS parameters for the connection, the SL protocol offers a number of services to the protocol such as integrity and confidentiality. In addition to the protection of the UDSES messages intended for either the FCNS communication layers adjacent to it or the peer entity, SL provides the necessary services for address verification and synchronisation procedures. UDSES issues the respective primitive whenever a connection is established or a message is received by the destination to verify that the nodes present in the session conform to a list identifying the valid network elements. Moreover, the SL decides on whether it can

support the session data transfer process and the requested QoS for the connection.

For the TX_LAYER protocol, security services include the identification and enforcement of the required mechanisms for message protection, both on an FCNS internal and end-to-end basis. The mechanism for the protection of the TX_LAYER messages sent to adjacent protocols follows the same principles of the context exchange process similar to the rest of the FCNS communication layers. As far as the end-to-end provision of security services is concerned, these are initialised during the handshake mechanism where the TX_LAYER protocol is informed of the keys and algorithms it should use to secure the peer messages. If authentication is requested, then the necessary mechanism should be enforced to create the message digest of the data to be sent, which should be identifiable only by the intended peer.

Similarly, the EE_LAYER is informed of the appropriate mechanisms that should be used for the internal and link-based encryption of the data packets sent between the network nodes. The process is again initiated via the handshake mechanism, although the algorithms and keys uploaded to the EE_LAYER instance concern the protection of the messages when traversing the intermediate nodes of the network topology. To minimise the possibility that any malicious hosts could intercept the message, the EE_LAYER informs the SL protocol of the route calculated for the connection and the formatting of the nodes present in it. The network operator should have been providing a temporary list of all routes present in the current network topology to facilitate the SL in performing the check routines on the routes sent by the EE_LAYER. If a node is listed as malicious or illicit, due to knowledge previously obtained by the network, then the route is rejected and the EE_LAYER is informed of the situation together with an action recommended by the SL. In any other case, the acceptance of the routing list signals its uploading to all nodes

present in the topology and its exchange with the intended destination for routing table updates.

Finally, the SL protocol has the task of providing the PHYS layer with the respective security mechanisms that should be used for the security of the FCNS frames prior to their transmission and the validation of the measures upon reception of the message. The appropriate functions are applied depending on user request, and, the FCNS provides the flexibility in supporting virtually any kind of data that may be in transit.

5.2.2.2 SL Data Transfer Monitoring and Management

The FCNS communication data transfer phase is signalled via the confirmation of the respective service primitive between the involved peers. The issues governing the security services afforded for the connection should have been negotiated during the connection establishment phase, with the SL protocol instructing the FCNS layers as to the keys and algorithms to be used for the messages protection.

The data transfer phase of the SL protocol actually forms a monitoring and management function of the layer, since no actual data is exchanged between the peer SL entities. Instead, the SL performs a series of actions used to facilitate the support of the requested QoS levels for the communication, both for the end-to-end peers, as well as for the intermediate routing nodes. These functions involve mainly the validation of the address fields present in the data messages, as well as error monitoring capabilities offered via the FCNSEP.

For the node instance it is recommended that whenever a new data block is to be transmitted the SL should check the security fields of the message forwarded, by obtaining a dummy data message from the application. The purpose of this check routine targets the early detection of any miscalculated authentication digests,

corrupted data and/or integrity and confidentiality features present in the data. It is up to the FCNS communication layers to ensure the appropriate enforcement of the security rules for the connection, yet, detection of implementation pitfalls may decrease the possibility of an undetected error from the network and a possible successful active attack.

The same principle applies for the UDPRES messages, where the receiving entity identifies whether the correct context extensions have been employed, that is, whether the transfer syntax has been effectively applied for the connection. If an error is detected at the UDPRES protocol, this is signalled to the SL in the form of a NODE_ALERT message, with the SL having the task of devising an appropriate solution for the recovery of the FCNS instance.

UDSES is also the protocol point where address validation procedures take place for every block of messages arriving at the receiver. Although the reliable transmission of the user and/or signalling data is left to the lower FCNS layers, UDSSES provides an extra measure in identifying the validity of the nodes participating in the connection. This measure comes into effect to verify that the messages have not been tampered with at the sending instance prior to their transmission. The list maintained by the SL should have been exchanged between the peers during the SC exchange and the handshake process, bearing information for the nodes present in that particular connection.

The TX_LAYER protocol addresses the issues of the end-to-end security for the peer messages by applying the necessary functions needed to support the required QoS. Following the validation of the message, the TX_LAYER should inform the SL of only the unrecoverable errors that might occur, in addition to any requests for SC renegotiation between the peers. The latter case should only come as a result of a detected message manipulation, where an adversary has allegedly compromised the keys used for the connection. In any other case, communication should

proceed based on the parameters agreed in the establishment phase. If re-synchronisation has to be applied to the data stream, then this is supported by the SL, which monitors the exchange of messages notifying the users of the support it can provide for the new connection factors.

Connectivity support is also offered to the EE_LAYER and PHYS layer, which constitute the subnetwork responsible for the routing and transmission of the data messages. Link-based encryption and frame security are the two functions for which the SL protocol has to afford services, in addition to error monitoring via the FCNSEP. Routing functions, including recalculation of the used path in the case of an error, should be signalled to the SL, which would have to verify the new route and nodes present on the path before notifying the peer of the alteration that has occurred.

5.2.2.3 SL Connection Release

Following the data transmission phase, nodes should confirm the appropriate request for the orderly release of the association set between them. Connection termination is signalled to the SL via the NODE_IDLE message by the UDPRES protocol, upon reception of the approval by the peer entity. The SL protocol will have to instruct the FCNS communication layers to return to their idle state for the particular application instance, an action achieved via the SET_STATUS message.

If the operator has requested the implementation of traffic padding mechanisms for the specific topology, then the FCNS stack should produce random data that would be secured as if it were normal data and sent along the same path data messages have been routed through. In this case the security parameters used for the bogus messages should remain unaltered until another connection request has been made, where new contexts should be agreed before terminating the traffic padding mechanism. On the other hand, if the operator decides upon maintaining clear

paths in the topology, then the SET_STATUS message should instruct the FCNS communication layers to reside in their idle state, waiting for a new connection request to be made.

In contrast, disorderly release should result in the immediate termination of the connection, which may include the discarding of any pending data messages or service requests. Traffic padding procedures should not be supported, unless there exists a specific context set for this implementation, since contexts allocated for the connection should have been released upon reception of the COMM_ABORT message signalling the termination.

5.3 FCNS Keystream Generator

The FCNS keystream generator is a pseudorandom number generator (PRNG) structure used to produce the secret keys necessary for the protection of the FCNS internal messages. It is based on the multiply-with-carry generators (MWCG) [152], providing periods of the order of 2^{32} and above before repetition.

Its design has been based on the fact that the keys used to secure the data messages should be different to those used for the exchange of primitives and be independent of the network on which the FCNS is implemented. As an example, the UMTS CN provides for the user and network authentication via its security functions offered by the Home Location Register (HLR) entity. These should not be used for the encryption of the messages to be transferred between the CN network elements or the layers of the FCNS stack, since that would increase the possibility of a successful attack on the network protocol in case the attacker obtains information as to the keys used for the data protection.

The following subsections outline the architectural and functional view of the FCNS keystream generator, as well as the statistical and cryptographic tests the

generator has undergone to provide a measure of the security services it can provide FCNS with.

5.3.1 Description

FCNS keystream generator uses three MWCGs basing its operation on three different internal states, as depicted in Figure 5.4, increasing its protection against cryptanalytic attacks. The seeding of the generators can either be done by the user, by the operator or by the protocol in the form of a list containing numbers acting as possible seeds for the machine. In the first two cases, appropriate security measures are clearly required to transfer and store the keys.

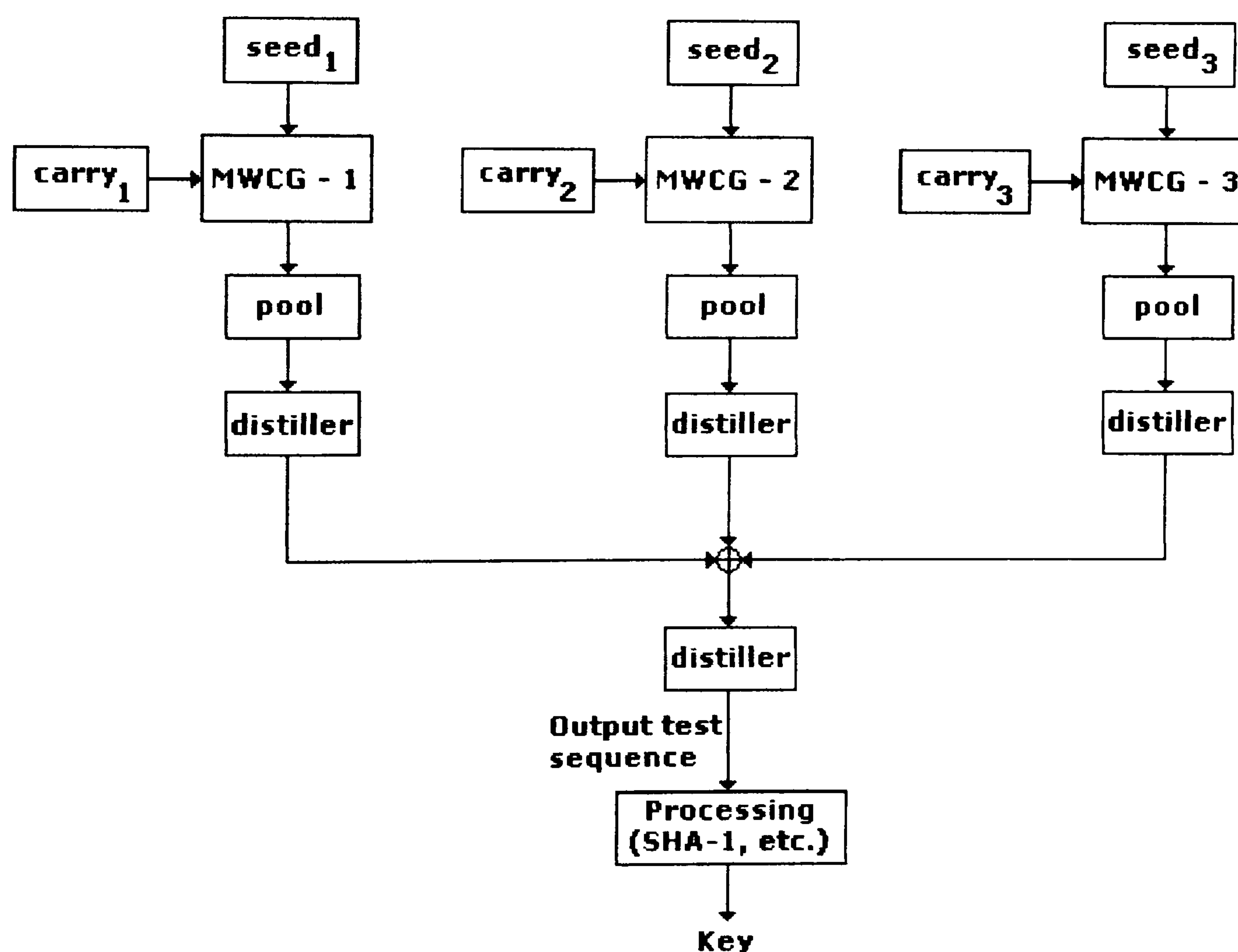


Figure 5.4: FCNS Keystream generator architectural view

Each of the individual generators accepts an initial seed and carry value, which will be used for the consequent calculations of the random sequences. These output sequences are then used to form an entropy pool, whereby a distilling function, such as the SHA-1 hash algorithm [153], should be employed to remove any

correlation and bias in the numbers obtained. The numbers are then fed into an XOR operation creating the output sequence used for the FCNS keystream generator statistical measurements, which in turn is passed onto the final SHA-1 function to create the 160-bit FCNS key.

Other options for the processing of the output pseudorandom stream include algorithms such as RC5 [154] or MD5 [155]. SHA-1 currently constitutes the state-of-the-art hash function in that it provides a 160-bit long key and has still to be successfully cryptanalysed.

The calculations of the values from each generator have been based on the following random number generators:

- Simple MWCG

$$x_{n-1} = a \times s + b \bmod(m)$$

Equation 5-1

$$x = ax_{n-1} \oplus bx_{n-1} + carry \bmod(m)$$

Where s is the initial seed value and m should be 2^{32} .

- Recursion with carry (RWC) - MWCG extension

$$x_n = a_1 x_{n-1} + a_2 x_{n-2} + \dots + a_r x_{n-r} + carry \bmod b$$

Equation 5-2

$$m = a_r b^r + a_{r-1} b^{r-1} + \dots + a_1 b - 1$$

The length of the generator will be the order of b for the modulus value m given above [156].

For the FCNS architecture, the example generators have been built to test their performance in producing pseudorandom number sequences. The design has been based to produce random sequences conforming to the specifications for security-purposed implementation of the generator, defined in the Federal Information Processing Standard (FIPS) 140-2 [157] and [158].

PRNG tests fall into two main categories depending on the purpose of their use in an operational environment. The first set of tests is classified as statistical tests [159], whereas a further step in addressing performance measurements of the generators for use in security applications is denoted as cryptographic tests. Both categories concentrate on specific factors and properties of the PRNG in question, enabling the comparison of the output sequence against the properties of a truly random one. The following two sections identify these categories and depict the performance of the FCNS keystream generator whilst subjected to both sets of tests.

5.3.2 Statistical Tests

The aim of these measurements is the optimisation of the generator to conform to regulations governing its use in security applications, such as the provision of secret keys for data and/or signalling security. In broad terms, the generator should experience [160]:

- Good distribution, in the sense that its output sequence should not be distinguished from a random sequence, given the unawareness of the algorithm used to produce the numbers.
- Long period, providing a minimum level of security against direct cryptanalytic (brute force) attacks.
- Repeatability, meaning that the PRNG should be able to produce a subsequence of the whole stream without having to produce the whole sequence of numbers.
- Portability to ensure that the generator can be implemented in any platform and that for the same input sequence, the same output one is generated, and
- Efficiency in accurately performing necessary calculations the algorithm is intended for.

The package DIEHARD [161] has been used to enable the identification of the FCNS keystream generator statistical properties and its conformance to the above requirements. Other packages include the pLab [162] and the NIST [163] test suites for PRNG structures, sharing similar characteristics to the DIEHARD battery, in relation to the tests provided for the generator output sequences. The selection of the DIEHARD suite was made on an ease-of-use and implementation basis.

This particular battery contains fifteen tests among which there are included those requested by the FIPS 140-2 recommendation for PRNG use in security environments. The proposal requires that 20,000 consecutive output bits be subjected to the following tests:

- The *Frequency (monobit) test* whose purpose is the identification of the proportion of 1's and 0's in the output sequence. The number of 1's and 0's should be counted and found to be between 9,725 and 10,275 for the test to be successful.
- *Poker test*, in which the stream is divided into 5,000 consecutive 4-bit segments. The number of occurrences of the 16 possible 4-bit values should be counted and stored, denoting as $f(i)$ the number of each 4-bit value i , where $0 \leq i \leq 15$. The test is passed if the number x , where

$$x = \left(\frac{16}{5000} \right) \times \left(\sum_{i=0}^{15} [f(i)^2] \right) - 5000 \text{ has a value between 2.16 and 46.17.}$$

- The *Runs test*, whereby the sequence of the 20,000 bits is inspected to identify the maximal successions of 1s and 0s in it. These successions are then measured to provide the number of 1s or 0s forming them, denoting the received value as a run. The test is regarded as successful, when the length, in number of bits, of the run conforms to the intervals presented in Table 5.2.

Table 5.2: Required length intervals for the Runs statistical test

Length of run	Required interval
1	2,315 – 2,685
2	1,114 – 1,386
3	527 – 723
4	240 – 384
5	103 – 209
6+	103 – 209

- Finally, the *Long Runs test* where the output sequence is checked against possible successions of 1s and/or 0s of length more than 26-bit long. The test is passed if there are no long runs present in the output under measurement.

In addition to those required by the FIPS 140-2 recommendation analysis, the FCNS keystream generator has been subjected to the following tests provided by the DIEHARD battery:

- *Birthday test*, where the generator sequence is checked against the level of occurrence between the output bits. The idea is based behind the birthday paradox used in cryptographic systems, where the probability that two people sharing the same birthday (normally 1/365) is increased together with the number of people present in a room [159]. For the FCNS keystream generator, the test provides a measure of the number of values occurring more than once in the list, detecting possible patterns of the output sequence that could minimise the generator's susceptibility to cryptanalytic attacks.
- *Serial (two-bit) test*, which represents a variation of the poker test, required by the FIPS 140-2 recommendation. In this case, the output sequence is divided into m-bit long overlapping sections and is checked as to the number

of occurrences of the $2m$ m-bit overlapping patterns. The idea behind this test is that the obtained number of occurrences should be approximately the same as would be expected from a random sequence.

The FCNS keystream generator has been successfully examined by all 15 tests of the DIEHARD battery, providing an adequate measure of the characteristics of the output sequence produced by the PRNG structure. The tests have been realised on an Athlon 900 MHz PC, running the Windows 2000 operating system with the FCNS generator being implemented under the Microsoft Visual C++ 6.0 environment. The selection of the seed and carry values for the three MWCG comprising the machine was made on a random basis, with the results of the above statistical tests being given in Appendix B.

Instead of providing the results of the statistical tests in the numerical form depicted in [157], the Figures of Appendix B illustrate the p values obtained by the experiments. These values represent the assumed distribution of the random variable(s) used for each test and should be uniform in $(0,1)$ for random sequences [161].

The applicability though of the FCNS PRNG in security environments has been further tested against proposed cryptanalytic attacks, which could lead to the compromise of the generator by an adversary. Although their use as a means of measuring the performance of such a generator is not officially recommended, they could be used to enhance the capabilities of the designer in identifying potential flaws of the developed system.

5.3.3 Cryptographic Tests

The cryptanalytic tests against random number generators have been suggested in [164] following the identification of possible attacks that could be launched against PRNG structures, in an attempt to manipulate the output sequence produced. Such attacks could be defined as the challenge in distinguishing between the pseudorandom output bits and those obtained by a truly random sequence. In a sense, the adversary would try and gain knowledge of the algorithm used to drive the generator, even if such information has been unavailable prior to the attack.

One of the most commonly launched attacks against any environment, either on a network or process level, is the *brute – force* attack, where the adversary performs an exhaustive search of the system parameters to disclose any information vital for its operation. In the PRNG context such an attempt is denoted as a *direct cryptanalytic* one, whereby the adversary is able to distinguish between the pseudorandom and random sequences with usually minimal effort. The attack is based on finding any particular patterns of the output stream, such as periodicity, which would imply the use of a deterministic mechanism in producing the numbers of the machine. The FCNS keystream generator, if correctly applied, should be immune to such attacks due to the long periods provided by the MWCG and RWC generators (order of 2^{32} and above) and the use of the one-way SHA-1 hash function acting as the distiller for the FCNS machine (Fig. 5.4). Any attempts at identifying the nature of the key produced should entail the thorough search of all possible combinations used for the stream production, as well as the cryptanalysis of the SHA-1 algorithm, something that has yet to be successful.

Another category of PRNG attacks encompasses the *input – based* attempts in which the attacker focuses on using knowledge or control of the PRNG inputs to predict the output sequences. The classification consists of the following three types of attacks:

- *Chosen – input* attacks, whereby the adversary attempts to manipulate the input of the generator usually provided by either the user or the system. Typical environments include the smart cards and other applications making use of passwords and identification numbers initiating a particular generator. The FCNS stack ensures the protection of the generator inputs, as long as the initial seeds and carry values are adequately secured in the databases containing this particular information, which should only be provided by the system operator.
- *Replayed – input* attacks, which are similar to the chosen-input attempts used to gain access and control of the generator output. The attack is successfully launched in cases where the user inputs the same password or same initial seeds feeding the generator, and the adversary monitors the communication to disclose the parameters in transit. The FCNS environment ensures the protection of the keystream generator by destroying previously used seeds and carry values, in an attempt to secure future communication requests and security contexts exchanged by the SL protocol.
- *Known – input* attacks, which can be applied when the attacker is able to know part or the whole of the input sequence used to initialise the generator. The particular attempt can be realised in software-based generators using operating system values, such as time parameters, since these could easily be monitored by the adversary. Protection against such attacks is similar to the previously described input-based types, with the FCNS generator being fed with values independent of the network or system environment on which the stack is realised.

The final category of PRNG attacks is denoted as the *state compromise extension* attempts, including:

- *Backtracking* attacks, whereby the attacker is able to use knowledge of a PRNG state compromised at a given time to disclose information for previous

outputs. Such an attack could be successful in cases where correlations and biases are identified in the produced sequence, wherein generation of the output bits of the generator is related to bits already produced. The FCNS generator distiller functions are used to provide a measure against the possibility that such an attack could be effectively launched against the system, resulting in the disclosure of the key generation mechanism.

- *Permanent compromise* attacks where all PRNG outputs are subjected to direct manipulation, provided that a certain state has been compromised at a given time. This particular attempt is an extension of the backtracking attack, with the generator been protected using the measure described above.
- *Iterative guessing* attacks, which could be launched by an attacker using knowledge of a secret state compromised at a given time to disclose a state at a later timer, even when the generator inputs cannot be identified. In a sense the adversary attempts to obtain information as to the internal states of the generator, even when an input-based attack has not been launched. A method used in the FCNS architecture to overcome such an attack is the distilling functions of the generator, which remove the correlations between the streams of each one of the three individual MWCG used forming the internal states of the machine, thus removing any dependence between them.
- *Man – in – the – middle* attacks, in which the attacker uses knowledge of a state at time t and $t + 2e$ to recover a state at time $t + e$. The attack is essentially a combination of the iterative guessing and backtracking attacks, as described above. Protection against these two types should result in securing the system against the man-in-the-middle one, although in this case the generator should be afforded functions protecting the internal states of each individual MWCG used.

5.4.4 Applicability

The mathematical background of the cryptographic PRNG attacks has fallen outside the scope of the thesis. Further information can be obtained in [159 - 169]. The scope of the FCNS keystream generator cryptanalysis has been the provision of an additional measure of security for the design of the stack architecture to the requirements imposed by the FIPS 140-2 recommendation. It could therefore be concluded that the generator should safely be used for the generation of the appropriate secret keys used in the SC exchange between the SL protocol and the FCNS communication layers, given that such tests should take place whenever new parameters need to be fed to the machine prior to its use in a new connection.

Chapter 6: Future Core Networks System Error Protocol (FCNSEP)

Chapter 6. Future Core Networks System Error Protocol (FCNSEP)

6.1 Overview

6.2 Architectural Analysis

6.3 FCNSEP Functionality

6.3.1 Interlayer Error Signalling

6.3.2 Peer Error Signalling

6.3.3 System Error Signalling

6.4 Security Considerations

6.4.1 Security Measures

6.4.2 FCNSEP and Currently Used Error Signalling Protocols

6.5 Applicability

Chapter 6 outlines the details of the functionality and implementation of the FCNSEP used for signalling error conditions between protocol stack elements and peer entities. The protocol forms part of the overall FCNS stack architecture and is initiated by the communication instances whenever a fault condition arises, to enable its correction and the continuation of the connection. FCNSEP can be used as a stand-alone error signalling and control system in packet-switched network architectures, provided that its messages are secured prior to their transmission via the communications channel. The chapter provides an overview of the system, followed by the architectural analysis of the FCNSEP depicting the messages format used to implement the protocol, as well as the actions suggested by the SL to the FCNS communication layers. Its functionality is then analysed with regards to the stack architecture, tracked by the security considerations and requirements for its deployment and applicability in network environments.

6.1 Overview

Error and control protocol architectures are essential to the adequate operation of a set of communication rules, whereby conditions that could prevent the normal communication flow are identified and signalled for correction. Their implementation usually follows the unsuccessful attempts of the system entity and/or process to recover from such a situation, consequently resulting in the necessity for external mechanisms to support and provide the required functions.

The parameters exchanged between peers are vital and hence a measure of protection should be available to secure the information in transit from passive and/or active attacks. FCNSEP has been designed to enable the protection of the data exchanged irrespective of the network running its instances. Security consequently becomes inevitable, and any messages sent and/or received unauthenticated and unencrypted should be ignored and discarded. By this method, FCNSEP ensures that at any time and under any protocol stack architecture, error control signalling is protected, to minimise the possibility that an attack could be successfully launched against the system.

The security context used for the FCNSEP messages should be different from those set for the peers and the FCNS communication layers. Although its implementation is supported by the protocols realising the connection, security forms an independent notion for the FCNSEP. The mechanism ensures that any security related information disclosure will not affect the FCNSEP operation or that of the data transfer process. At the same time, the mechanism removes any dependency of the signalling mechanism to the underlying protocol maintaining the particular communication.

FCNSEP is initiated by the SL protocol, upon reception of the error indication by an FCNS communication layer, as depicted in Figure 6.1. The NODE_ALERT message signals the error condition to the SL, which should reply by providing the

appropriate action required for the recovery and correction of the fault situation. If such an action cannot be offered, then the FCNSEP is used to advise the system of the circumstance and inform the protocol instances of the steps to be taken.

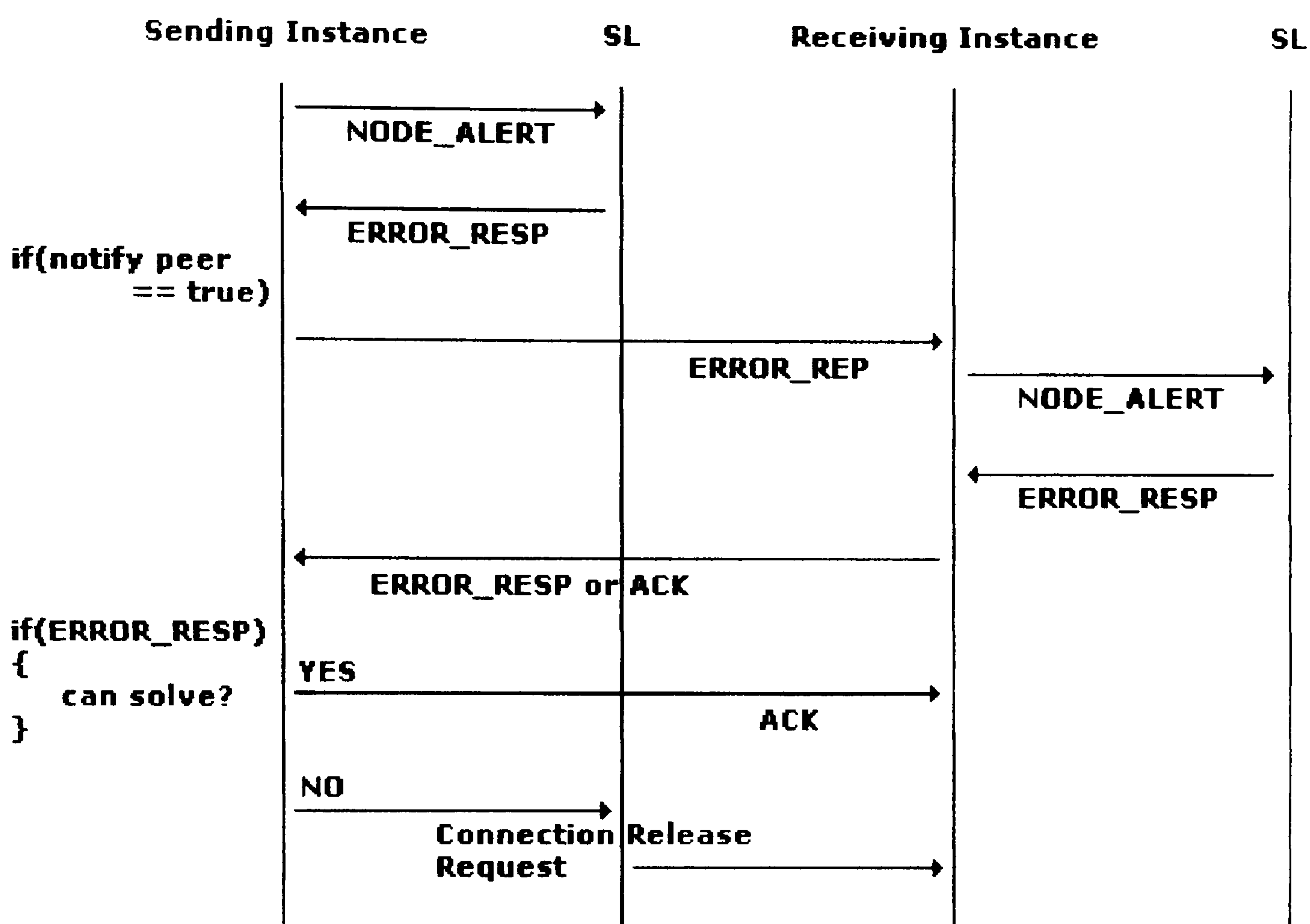


Figure 6.1: FCNSEP realisation in the FCNS stack architecture

On the other hand, if the necessary mechanisms can be made available by the SL, then FCNSEP is realised to provide the FCNS communication layers and/or the peer nodes with the respective indication. Depending on the layer that has detected the error, the appropriate security mechanisms should be enforced prior to the FCNSEP message transmission. This function enhances the identification of the layer for which the report is intended, in cases where due to a protocol error the message is forwarded to a layer different than the one intended. At the same time, the mechanism ensures that it is only the anticipated instance that could decipher the message, minimising the possibility of a successful disclosure attack at a certain FCNS communication layer point.

The following subsection identifies the messages used for the attainment of FCNSEP functions, providing also a description of the conditions signalling its initialisation.

6.2 Architectural Analysis

The FCNSEP architecture is based on the *ERROR_REP* and *ERROR_RESP* messages illustrated in Figures 6.2 and 6.3 respectively. They form part of the exchange procedure depicted in Figure 6.1 and are responsible for the transmission of the information representing the error condition and the action suggested for its recovery.

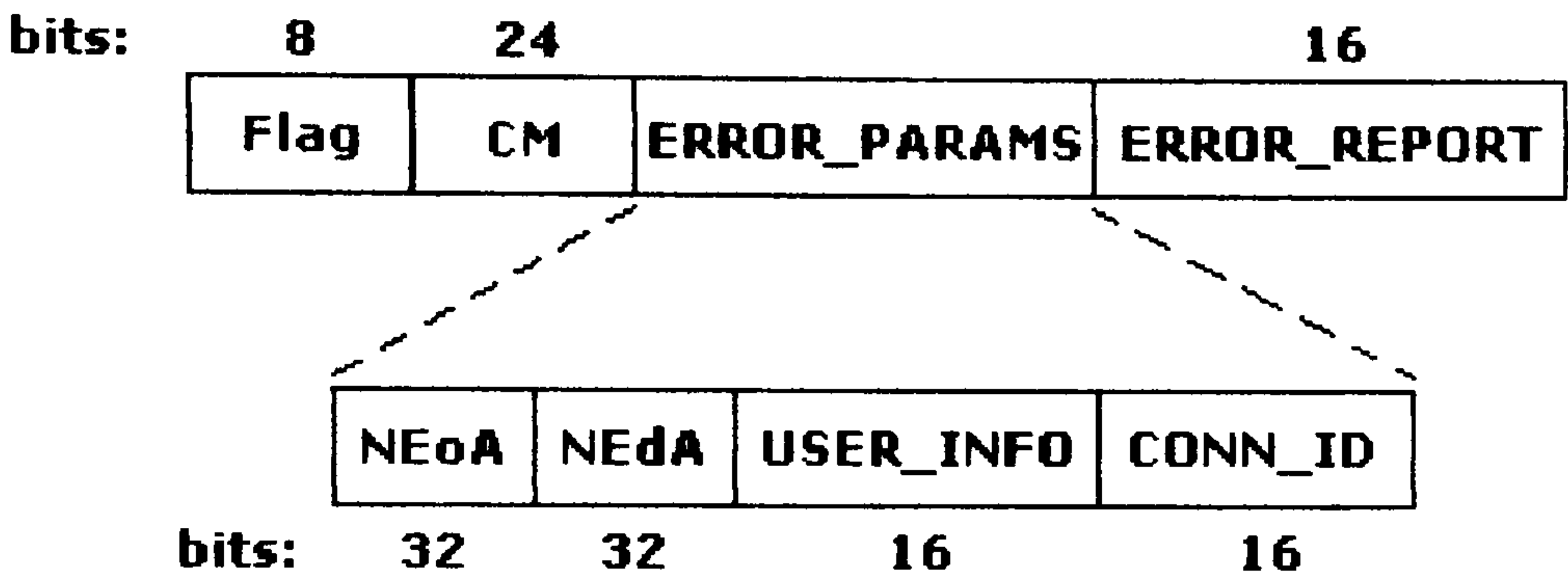


Figure 6.2: *ERROR_REP* FCNSEP message structure

For the *ERROR_REP* message, the following fields contain the necessary signalling information for the fault condition:

- *ERROR_PARAMS* field, which includes parameters such as the sending node and intended destination of the FCNSEP message, as well as information about the user and the connection on which the error has been identified. If the destination is not directly adjacent to the transmitted instance, then the message should be mapped onto an *EE_LAYER* packet and be encrypted with link-based mechanisms, to ensure the secured traversing of the *ERROR_REP* via the intermediate nodes. The *USER_INFO* and *CONN_ID* fields include details on the network connection upon which communication is based, as well as QoS parameters that should be used to identify the priority the SL should give the message upon its reception.
- *ERROR_REPORT* field, which is used to indicate the condition that triggered the FCNSEP message. Table 6.1 contains the possibilities for each of the FCNS communication layers, as well as for the SL protocol.

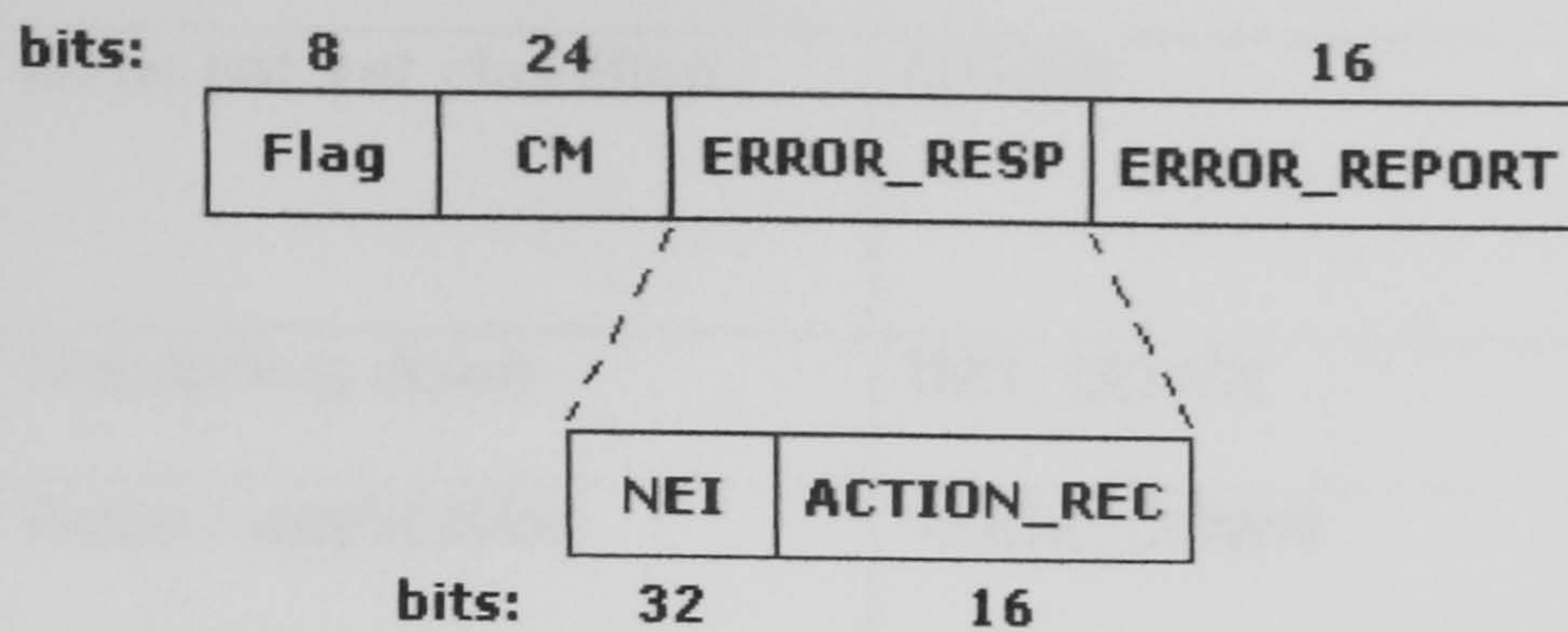


Figure 6.3: ERROR_RESP FCNSEP message structure

For the FCNSEP ERROR_RESP message, the fields containing parameters as to the action that the peer or layer should follow for the error correction are as follows:

- *ERROR_RESP* or *ERROR_RESPONSE* field that includes information about the network element or peer for which the action has been suggested (the *NEI* field), as well as the *ACTION_REC* field containing that particular action.
- *ERROR_REPORT* field, which is similar to the respective one of the *ERROR_REP* message and whose inclusion denotes the error for which the *ERROR_REP* and consequently *ERROR_RESP* messages have been sent. Failure in including such data in the message results in the discarding of the FCNSEP response and the indication of the error to the peer entity or the SL protocol.

The specification diagram of the FCNSEP architecture is given in Figure A.9 of Appendix A, including information as to the states governing the operation of the protocol. For every possible error situation there exists a list of suggested actions, to which the SL protocol should refer prior to choosing the most appropriate one for the connection.

Table 6.1: Error conditions for the FCNSEP ERROR_REPORT message field

Error condition	Representation	Layer(s) that could detect it
Fatal error	FATAL	All (leads to disorderly connection release)
A non-fatal error	NON_FATAL	All

Error not yet classified	OTHER	All (system signalling imperative)
Network is down	NET_DOWN	TX_LAYER, EE_LAYER, PHYS
Node / application instance is down	NODE_DOWN	UDPRES, TX_LAYER, EE_LAYER, SL
Link failed	LINK_DOWN	EE_LAYER, PHYS
Link is congested	LINK_CONG	EE_LAYER, PHYS
Node instance is busy / cannot accept request	NODE_BUSY	UDPRES, UDSES, TX_LAYER, SL
Wrong padding mechanism	PAD_BAD	TX_LAYER, EE_LAYER, PHYS
Wrong checksum value	CHECK_BAD	TX_LAYER, EE_LAYER, PHYS
Wrong security header	CM_BAD	All
Wrong timer	TIMER_BAD	UDPRES, UDSES, TX_LAYER, PHYS
Message header corrupted	HEADER_BAD	All
Timer is too high	TIMER_HIGH	UDPRES, TX_LAYER, PHYS, SL
Timer is too low	TIMER_LOW	UDPRES, TX_LAYER, PHYS, SL
No ACK received	NO_ACK	UDSES, PHYS, SL
No NAK received	NO_NAK	UDSES, PHYS, SL
ACK message corrupted	ACK_BAD	All
NAK message corrupted	NAK_BAD	All
Route calculation procedure failed	ROUTE_CALC_FAIL	EE_LAYER, SL
Segmentation / reassembly failed	SAR_FAILURE	UDSES, TX_LAYER, EE_LAYER
Continuously received duplicates	DUPLICATE_MSG	All
No such node present in	NO_SUCH_NODE	UDSES, EE_LAYER, PHYS, SL

the route		
No such link present in the route	NO_SUCH_LINK	EE_LAYER, PHYS, SL
No subnetwork to support the connection	NO_SUBNET	EE_LAYER, PHYS
Subnetwork is down	DOWN_NODE	TX_LAYER, EE_LAYER, PHYS
Node classified as suspicious	NODE_SUSPECT	EE_LAYER, SL
Node cannot fulfil request (for the subnetwork)	NODE_NOT_CAPABLE	TX_LAYER
Hardware error	ERROR_NODE	EE_LAYER, PHYS, SL
Protocol procedural error	PROT_ERROR	All
Destination unreachable	DEST_UNREACH	EE_LAYER, PHYS
Security / security context error	SEC_ERROR	All

Each of the error conditions can be identified, if such capability exists, by most of the layers of the FCNS architecture. As Table 6.1 indicates there may be cases where an error can only be detected by specific layers, since those errors may not affect the operation of the rest of the protocols. Although there is no need for a clear distinction in parameterising the network or protocol error conditions appearing in Table 6.1, it is imperative that the FCNSEP actions should differ for the interlayer and peer layer signalling, as described in the subsection below.

6.3 FCNSEP Functionality

The responsibilities of the FCNS error protocol can be summarised as follows:

- Interlayer error signalling
- Peer error signalling
- System signalling for unrecoverable and unidentified errors

These functions form the very essence of the FCNSEP and are used throughout the various phases of the communication. Since FCNSEP represents an integral part of the FCNS architecture, its use and initialisation should be independent of both the protocol phase and the FCNS state.

6.3.1 Interlayer Error Signalling

Following the reception of the initial connection request by the application instance, the node running the FCNS stack proceeds to establish a functional set of communication rules, to enable the necessary services requested for the connection. The FCNS protocols transition to a processing state, whereby the appropriate service access points are made available for accepting the respective primitives for the particular application.

Each of the FCNS layers forms a valid protocol, which is used to provide the suitable services to its user, virtually communicating with the peer entity at the same level (Chapter 4). A protocol could be used as a stand-alone set of communication rules, without the node running the entire FCNS stack. In such cases, the FCNSEP only provides peer entity error signalling capabilities due to the lack of a complete functional stack.

The FCNSEP self-sufficiency is thereby limited by the operation of the FCNS stack as a whole, in which case the means of signalling error conditions to the protocols of the FCNS structure becomes imperative. A protocol procedural error for example in the subnetwork layers, such as the EE_LAYER and the PHYS, could render the connection establishment or data phase impossible. Hence, the FCNSEP should be used to enable the upper FCNS layers notify the node of the fault condition and at the same time allow the system to assign another network connection for the particular session.

Usually the FCNSEP implementation is subject to the reception of a message in error for more than three consecutive times, or the failure in establishing certain connection parameters after an equal amount of attempts. This feature is used to minimise the possibility that FCNSEP functions come into effect for situations where error recovery may be achieved at no extra cost by other FCNS mechanisms. The FCNSEP is implemented only if this is regarded as an absolute necessity by the system or the node process, to enable the suitable use of the available link resources for the data transfer phase only.

When error signalling takes place, the protocol detecting the error should inform the SL of the fault condition, to provide the suggested for recovery action. If the SL protocol can offer the required action supporting the error correction and recovery procedure, then this is indicated to the layer via the ERROR_RESP message, including the reason for which the message has been sent to distinguish between any other NODE_ALERT requests made. Table 6.2 includes some of the possible values the ACTION_REC field (Figure 6.3) might take, notifying the FCNS protocol of the mechanism and procedure that should be followed to correct the error.

Table 6.2: Sample suggested error correction and recovery procedures

Error signalled	Action recommended	ACTION_REC value
NET_DOWN, NO_SUBNET, DOWN_NODE	Recalculate the route chosen	RECALCULATE_ROUTE
LINK_DOWN, LINK_CONG	Follow an alternative route from the list	FOLLOW_ALTERNATIVE
DUPLICATE_MSG, CM_BAD	Abort communication, error cannot be recovered from / action not provided	ABORT_COMM
TIMER_HIGH	Minimise timer value by t msec	MINIMISE_t

TIMER_LOW	Maximise timer value by t msec	MAXIMISE_t
CHECK_BAD	Apply another checksum calculation procedure	RECALCULATE_CHECK

Depending on the severity of the error, the SL protocol decides upon the action to be taken, including the release of the connection in cases where the layer cannot recover from the error condition. At the same time, it should also decide whether the user of the layer protocol should be notified of the condition or even the signalling of the situation to the peer.

Upon reception of the ERROR_RESP message, the respective FCNS communication layer has to check whether it can support the functions required to recover from it. If this is not possible, then the NODE_ALERT message is sent again, indicating the protocol's inability in conforming to the SL dictations. The node should keep a record of the number of error indications sent to the SL and if the number exceeds the predetermined threshold for the particular protocol (usually three tries), then the connection should be released and the system be notified of the situation following the signalling of the condition to the peer.

If the suggested action can be supported, then the necessary functions should be enforced, to correct the error that has occurred and to proceed in supporting the particular connection. If, for example, a link has been found to be heavily congested resulting in delaying or discarding data packets, then an alternative route should be followed, as calculated during the connection establishment process. If such an alternative does not exist, then the layer should indicate the situation to the SL, which should in turn advise the EE_LAYER in re-calculating the route for the forwarding of the data messages. If the re-calculation process yields an alternative or such is provided by the network operator, then the routing tables of the nodes present in the network topology should be updated accordingly and

the peer notified of the alteration occurred. In contrast, in cases where route calculation might not take place, then the SL protocol should instruct the peer node in providing the suitable capabilities else signal the FCNS layers in releasing the parameters and security contexts set for the connection.

Finally, it may be the case that the SL is unable to provide details about the correction and recovery from the fault condition. In such situations, the NODE_ALERT message is replied with a NAK message, signalling the notification of the system of the condition arisen. Care should be taken in dismissing any error indication by the layers for fear of an active attack by an adversary, whereby illicit error request messages are inserted to disrupt the connection service or obtain information about the operation of the SL protocol. It is therefore imperative that all FCNSEP related messages are secured using the parameters exchanged via the respective SC during connection establishment phase.

If the error can be corrected locally, then communication will proceed as required without the peer being notified about the condition, a measure used to avoid redundant information in the network channels. On the other hand, peer error signalling should be enabled if the condition's severity is such that either the fault should be corrected at the peer or the connection becomes extremely difficult to support.

6.3.2 Peer Error Signalling

FCNSEP peer error signalling is a process initiated upon indication from the SL protocol that the peer should be notified of the error condition. The indication should be included in the ACTION_REC field of the ERROR_RESP message, as well as providing the appropriate NEI identifier in the respective field. The process involves the transmission of the ERROR_REP message towards the peer entity involved in the error condition. The message could be initiated by either the data

messages intended destination or an intermediate router responsible for the forwarding of the data to their recipient. Typical examples include the indication of an ARQ or TX_LAYER flow control failure, the notification of an unreachable host or the detection of a congested and/or faulty transmission link.

Upon reception of the ERROR_REP message, the entity should perform a series of tests aiming at identifying any security inconsistencies that might be present in the FCNSEP message. Table 6.3 depicts some of the possible errors that could be detected on the ERROR_REP message.

Table 6.3: ERROR_REP erroneous reception conditions

Error condition	Signalled value
Message header and/or data received corrupted	CORRUPTED_MSG
Faulty security headers	WRONG_CM
Wrong destination address (e.g. node not involved in the communication)	WRONG_ADDRESS

These typical error values are used to enable the creation of the NAK_REP message depicted in Figure 6.4 below, and are contained in its REASON_REJ field.

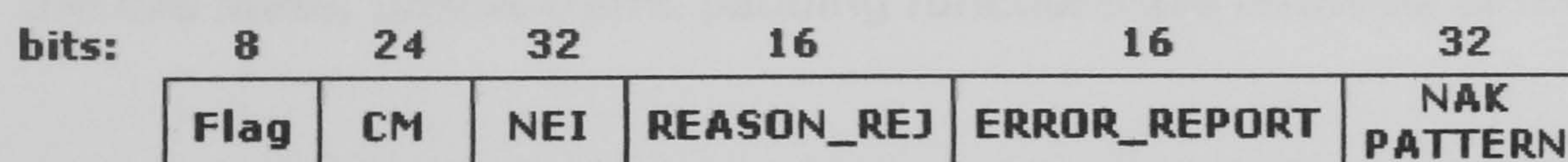


Figure 6.4: NAK_REP message structure

The ERROR_REPORT field contains information enabling the recipient of the message to identify the exact ERROR_REP message received on error, whilst the NAK_PATTERN field holds a unique pattern used to offer an extra degree of protection of the message against possible manipulation and replay attacks.

If the ERROR_REP message is successfully authenticated and decrypted, then the peer notifies the SL protocol of the error condition, with the process involving similar steps to the interlayer error signalling procedures.

If the layer entity can, in conjunction with the SL protocol, identify and correct the error, then an acknowledgment is sent back to the peer, to indicate that communication should proceed. Any suggested actions related to the receiving instance are mapped onto the data messages, reducing the overhead that would be produced by a further ERROR_RESP message transmission. In contrast, if further negotiation of the necessary parameters needs to take place, then this is signalled via the ERROR_RESP message to the appropriate node or the system.

Upon reception of the ERROR_REP response, the node enters a final checking routine phase, wherein the layer attempts to overcome the problem with the action proposed by the SL. If the fault cannot be corrected, then the node should initiate the connection release phase by sending the appropriate service primitive request. The SL notifies the system of the condition that led to the termination of the association, so that appropriate actions can be found and uploaded to the node instances accordingly. Any parameters and variables set for the specific connection are released from the node's memory and the FCNS communication layer is reset to the idle state, unless traffic padding functions are afforded to the network topology.

6.3.3 System Error Signalling

System error signalling procedures involve the explicit notification of the system network elements in cases where error recovery services cannot be offered to a particular FCNS instance. It usually follows the unsuccessful attempts of the SL protocol to negotiate with the FCNS communication layer and/or the peer entity the appropriate mechanisms that could be used for the error correction phase of a particular connection.

The error notification procedure is initiated by the SL protocol instance in the form of the ERROR_REP message, where an indication of the fault condition is given, together with identification of the particular instance that experienced the situation. If the network operator can provide the necessary functions for the correction and recovery procedures, then these are included in the ERROR_RESP message sent back to the SL. If not, the system informs the SL protocol that it is initiating the connection release process, signalled to the FCNS communication layers by the SET_STATUS message. An entry of the condition that caused the termination of the association should be kept in a file, to enable the monitoring of the situation and the development of an adequate function that could be used if such situations may ever arise.

Upon reception of the ERROR_RESP message, the SL protocol should be able to construct a suitable recommended action for the layer entities and forward this on another ERROR_RESP message. It is important that the mechanisms used to secure the system signalling messages be independent of those used for the interlayer and peer error notification functions to protect the network against possible active attacks.

The principles governing the detection and signalling of error conditions to the system entity follow those presented for the peer error notification case, since they involve the communication of a particular FCNS instance with an external network or peer element. The only difference observed in the two processes concerns the uploading of the information obtained to all nodes present on the given topology, increasing the network elements awareness of any fault conditions that might affect their operation. In contrast, peer error signalling is a locally based solution where mechanisms are addressed only to the nodes involved in the data transfer communication process.

As a final remark, it should be noted that the purpose of the FCNSEP is the signalling of error situations to peers and the negotiation of the proposed schemes for their correction. The possibilities and fault conditions that may occur in a network vary from software to hardware faults. FCNSEP implementation does not dictate the mandatory correction of and recovery from such situations, though it provides an adequate and secured method in responding to system calls regarding the maintenance of the requested connection QoS.

6.4 Security Considerations

Message security is a notion strongly supported in the FCNSEP architecture, since all messages involved in the error signalling procedure should be protected prior to their transmission. As described in the previous sections, if an FCNSEP message is received unsecured, it is ignored and discarded, irrespective of the error condition in the network.

The mechanisms used to protect the ERROR_REP and ERROR_RESP structures, including the independent secret keys previously discussed in the content of the other layers, are exchanged and agreed between the FCNS communication layers and the peers during the SC exchange process.

6.4.1 Security Measures

The SL protocol provides the required services for the FCNSEP signalling messages. The service of the FCNSEP can therefore be regarded as connectionless – based, in the sense that the messages can be transmitted by an FCNS communication layer towards the SL at any time, no matter the connection which the layer might be involved with. Similarly, a peer entity might be supporting more than one communication processes, yet an FCNSEP request could be launched at any time irrespective of those processes and the phase association would reside at that time.

Although the appropriate connection parameters are included in the FCNSEP messages to facilitate the identification and correction of the fault condition, the end-to-end connectivity issues are left to the FCNS communication layers responsible for the actual transmission of the error signalling information. The connection-oriented functions supporting the association should have been negotiated during the connection establishment process between the peer entities.

Additionally, the security contexts agreed, provide for the peer entity authentication, integrity and confidentiality services to be used throughout the communication process, securing the data transfer on a connection-oriented rather than on a message-based foundation. Consequently, FCNSEP is secured on a connectionless service basis, to enable the protection of the error signalling data irrespective of the connection in question, since the peers have already been afforded the required services during association set up. However, for the FCNSEP peer error signalling process, the network operator is able to request peer-based security services for the connection, given that the appropriate certification authority can provide the necessary parameters for the FCNSEP protection.

In general situations, it is recommended that since the FCNSEP messages would be mapped into the FCNS communication layer connections, that they be afforded message-based security services, to eliminate to a great extent the need for an external authority providing for the appropriate security parameters.

One of the first security measures applied for the FCNSEP messages is the authentication of the data transferred either between the FCNS layers or the peer entities. The mechanisms, other than the secret keys, could be the same for both cases. The message digest is included in the data, where an identifier is also contained in the message for protection against possible replay attacks. The identifier must be specific for a connection or error-signalling request and never be used for another FCNSEP initiation.

Message authentication services follow the data origin verification principles of the security functions offered by the SL protocol. The peer entities confirm the validity of only the sender of the FCNSEP message, relying on the peer-entity authentication functions to confirm the legitimacy of the peers themselves. The source and data recipients verify that the message can indeed traverse the network and is not the product of an illicit node attempting to manipulate the connection.

Since the validity of the message can only ensure the legitimacy of the transmitter, FCNSEP messages are afforded the appropriate integrity functions to certify that their contents have not been tampered with, either by an adversary or a faulty transmission link. Encryption of the message is mandatory even in the interlayer signalling cases, to counter an active protocol attack.

If an attacker were able to alter the information contained in the FCNSEP data, then that could cause serious problems possibly leading to disconnection. A typical example of such a case is the repeated manipulation of the ERROR_REP and ERROR_RESP messages, whereby the adversary alters the contents of the ERROR_REPORT field issuing a FATAL error request to the peer entity. If the node accepts the request as being a legitimate one, then that would lead to the termination of the connection. Therefore, a successful DoS attack could be mounted if the messages are not securely sent over the communications channel.

Furthermore, since the messages are intended for a particular destination, the FCNSEP security mechanisms can additionally ensure the confidentiality of the data in transit, together with verifying its integrity. Message integrity functions may be coupled with a public key scheme, whereby only the legitimate destination is able to decipher the message.

Application of functions such as non-repudiation and access control are not of importance for the FCNSEP architecture. The former is rendered redundant by the

monitoring and maintenance capabilities FCNS can provide to an already established connection. The latter is made unnecessary by the assumption in this work that all nodes are peers.

The security implications of the FCNSEP architecture form one of the essential elements of the protocol and any unsecured messages are discarded. The information transferred in the error signalling messages is too vital to be sent in plaintext format, with any possible manipulation leading to potential DoS attacks against the connection.

6.4.2 FCNSEP and Currently Used Error Signalling Protocols

Error and control protocols have always been used in computer and telecommunication systems as a means of maintaining the required QoS for a connection, by providing peers the appropriate information for the communication well being and recovery from situations that could affect its lifespan.

This session provides a comparison of the FCNSEP in relation to the Internet Control Message Protocol (ICMP) architecture of the IP family. ICMP is currently in its sixth version [170], following the modification of the IP specification and its transition to IPv6, and is forming one of the most commonly used error and control signalling mechanisms. Figure 6.5 illustrates the general structure of the ICMPv6 message header, including a list of the possible errors that could trigger its operation.

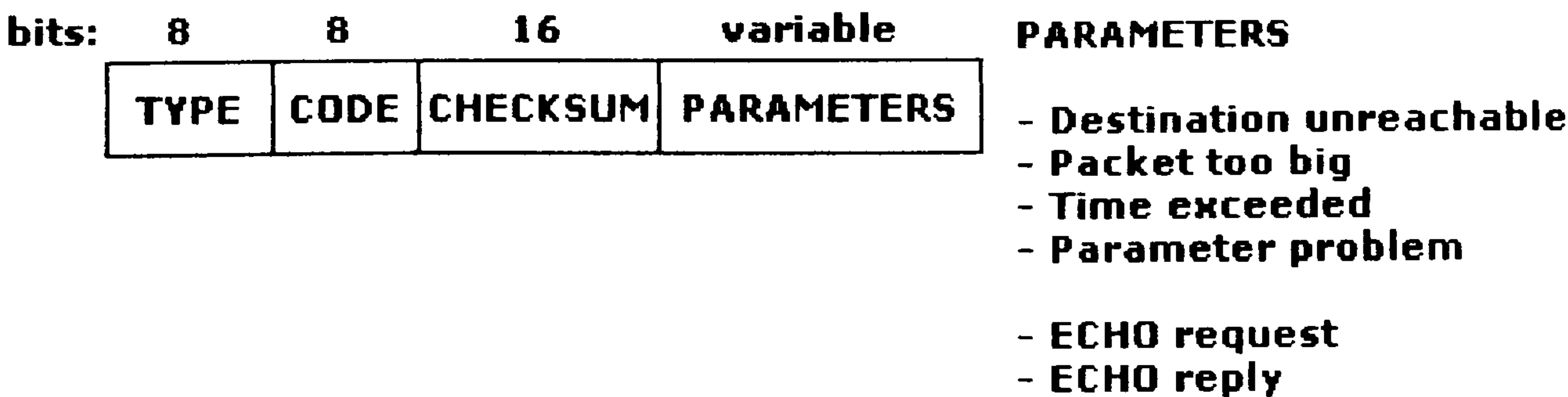


Figure 6.5: ICMPv6 header message structure

Depending on the fault situation, the *PARAMETERS* field is updated accordingly, including an amount of the data sent when the problem has been detected, to enable the message recipient identify the location of the fault condition in the data stream and the upper layer(s) involved in the situation.

ICMP is used in the TCP/IP protocol stack as the architecture responsible in ensuring that the data messages conform to the parameters set during the connection establishment phase and also that their routing towards the data's intended destination is successful. The protocol is not responsible for end-to-end connectivity issues, usually handled by the TCP flow control mechanisms, and hence can provide notification for only a limited number of error conditions, as depicted in Figure 6.5.

Consequently, the responsibilities of ICMP are somewhat limited to congestion handling, either as a congestion avoidance or congestion recovery mechanism. In the first case the purpose of its initiation is the avoidance of a zero throughput network appearance. The second case involves issues of congestion recovery even when such measures have already been enforced on the particular connection. A typical method realising the congestion monitoring and maintenance services is the *source quench* process, whereby the source is instructed to reduce the rate at which information is forwarded onto the network topology. Unfortunately, ICMP does not inform the end system of the actual reason for which the request has been launched, and hence the indication may not always be specific to the particular system, in the sense that it may not imply the signalled entity being the cause of congestion.

ICMPv6 also provides a measure of informing the data source of any inconsistencies in the network topology that may deny the transmission of its messages towards the receiver. The *destination unreachable* ICMP message includes information as to whether a route to the peer actually exists and/or details about the legitimacy and

capabilities of the node directly connected to the sending instance. Table 6.4 indicates the possibilities available for the notification of the sending instance with respect to the status of the data recipient, as set by the specification of the protocol.

Table 6.4: Destination unreachable ICMP notification instances

Destination unreachable	<ul style="list-style-type: none"> • No route to destination • Administratively prohibited • Not a neighbour • Address unreachable • Port unreachable
-------------------------	--

The ICMP protocol forms an integral part of the IP stack implementation, resulting in prohibiting its use to architectures based on another protocol stack rather than IP. Until the design of the IP security architecture (IPsec), all ICMP messages were sent in cleartext format, leaving the network susceptible to attacks by unauthorised parties [171]. Although IPsec provides for the protection of the protocol's messages against modification and DoS attacks, the functions of the architecture closely bind any security considerations to the IPsec structure for which security is not always imperative. This implies that ICMP messages could still be sent in plaintext format, if the association was to be initiated without any protection mechanisms enabled.

The provision of the IPsec services for the ICMPv6 messages should ensure the adequate protection of the protocol data against unauthorised modification and information disclosure attacks. This notion does not imply any protection of the system against attacks aiming at the IPsec architecture itself, by exploiting vulnerabilities of the system running the IPv6 protocol [172] and [173].

The overall security of the ICMP system relies on parameters external to the protocol and set of communication rules, in contrast to the FCNSEP where security services are afforded for the protocol independently of the underlying network structure and FCNS implementation. Moreover, FCNSEP addresses a wide variety of error conditions that could affect the communication flow, reporting any condition to the system and/or the peer entity and not only those specific for the link-based data transmission. Finally, FCNSEP is not bound by the functionality of the FCNS stack and hence can be used in virtually any packet-switched architecture and telecommunication systems such as the UMTS CN. The following subsection identifies the areas of applicability of the FCNSEP with respect to the scope and notion of a possible FCNS network implementation.

6.5 Applicability

FCNSEP defines a framework for use within network architectures, where explicit error notification should take place irrespective of the technology supporting the connection. A typical example of the applicability of the FCNSEP is a generalised network model, where the node initiating the connection may be the subscriber of a UMTS network, requesting a data transmission connection with a node residing behind a packet-switched network. Figure 6.6 provides such an example, where the Internet network present between the UMTS CN and the particular node adds to the particularity of the topology, where various erroneous conditions may arise due to the diverse nature of its elements.

The sample network outlines a broad integrated topology, as such could be defined for a third generation telecommunication system, merging existing technologies to enable the communication between the mobile subscriber and Node B. The FCNS protocol stack could be used to securely transfer the required data via the elements of the UMTS CN, denoted as NE1 and NE2, and route the messages on the Internet network through the routing elements indicated as NE3, NE4 and NE5.

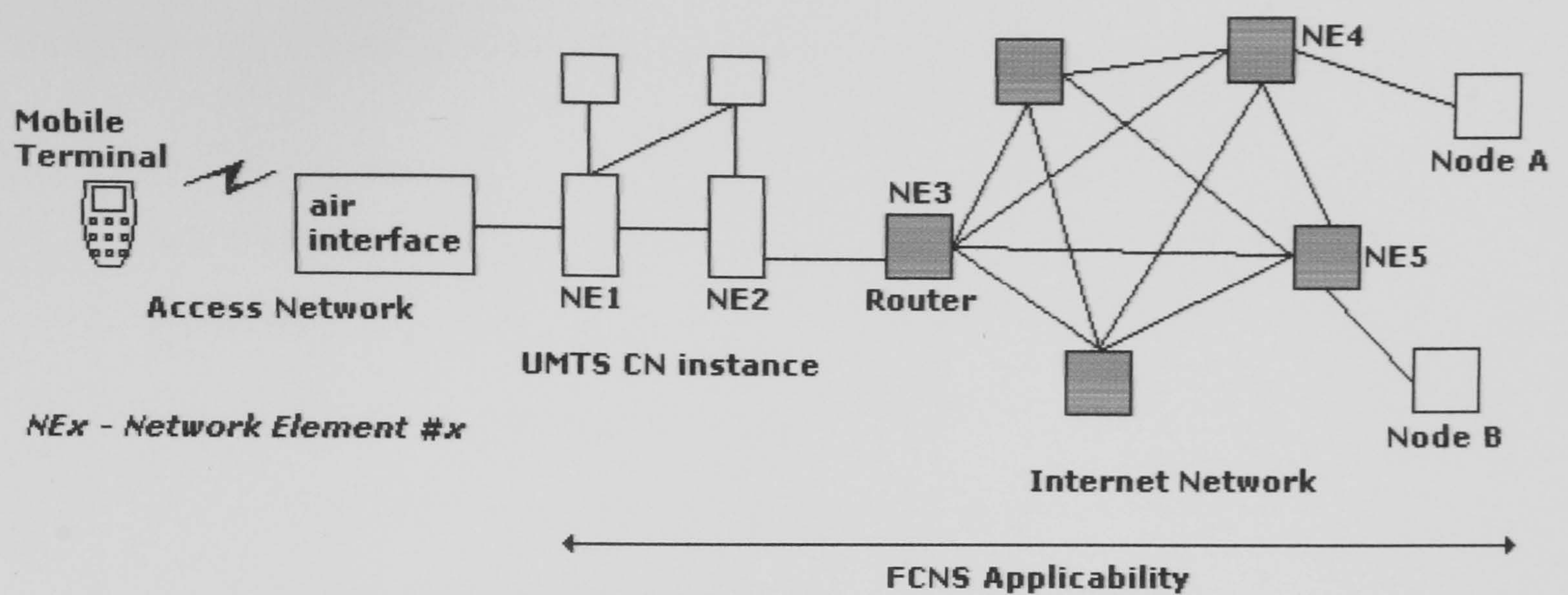


Figure 6.6: Sample integrated communications network

At any given time, an erroneous situation such as a faulty transmission link or excessive congestion could arise in any of the different network systems forming the example of Figure 6.6. This mandates the need for an architecture that could provide for the signalling of the error parameters throughout the environment nodes. FCNSEP can support the notification of the network elements involved in the communication process, irrespective of the nature of the network where they reside, thus offering a degree of connection monitoring throughout all phases of the association. The support of various architectures that could shape the set of connections in question renders FCNSEP a suitable mechanism for error signalling procedures, offering at the same time a degree of protection against active network attacks aiming at exploiting the valuable information exchanged between the peers.

The use of FCNSEP constitutes a last resort in attempting to notify and recover from an error network situation. Parameters such as congestion control could be added at the user frames or packets, to reduce the overhead that could be produced by the FCNSEP messages. However it is imperative that there exists a means of alerting the system administrator of any situation that could endanger the communication process, enabling not only the correction of such a situation but the initiation of the appropriate mechanisms that could afford its future prevention.

Chapter 7: Future Core Networks System Implementation

Chapter 7. Future Core Networks System Implementation

7.1 Overview

7.2 FCNS Design Process

7.2.1 Service Specification

7.2.2 Environment Parameters and Assumptions

7.2.3 Protocol Vocabulary

7.2.4 Message Formats

7.3 FCNS Prototype

7.3.1 Prototype Building

7.3.2 Validation Tool Applicability

7.4 Source Code Implementation

7.4.1 FCNS Stack Simulation Environment

7.4.2 Simulation Tool Applicability

In Chapter 7 an analysis of the FCNS design and implementation process is provided, which enabled the completion of the FCNS specification. A description is given as to the assumptions made for the environment in which FCNS would run, including details of the rules followed to realise the protocol. Furthermore, a presentation on the FCNS prototype building is provided, encompassing information for the validation tool used and its applicability. Finally, the FCNS source code implementation is analysed, with details on the simulator and the environments created for realising the proposed architecture.

7.1 Overview

Protocol design aims at identifying the necessary parameters used for the realisation of a given architecture. Available techniques target the establishment of an agreement about the usage of shared resources in a network of peers [6]. Current literature on protocol design issues represents the steps that can be taken to define the minimum functionality and services required for a protocol to function [174 - 176].

The definition of the set of communication rules governing a protocol, focuses on three main requirements [6]:

- The *syntax*, where definition is given as to the format of the valid protocol messages
- The *grammar*, which defines the procedure rules for the data exchange
- The *semantics*, forming the vocabulary of valid messages that can be exchanged

These rules encompass most of the details concerning the connection establishment, data transfer and termination process issues negotiated between the communicating peers. The validity and common understanding of the nodes present in the connection is vital for the operation of the protocol and consequently the connection realisation.

Nevertheless, conformance to the above requirements is only the first step in properly defining the protocol architecture. Specific elements of the structure must be defined to enable these requirements to be set. These elements are as follows [6], [174]:

- *Service* element, which represents the services to be offered by the protocol
- *Assumptions* taken for the environment protocol will be run
- *Vocabulary* of the messages used to implement the protocol
- *Encoding* for the specific messages

- *Procedure rules*, governing the message exchange process

The respective elements and requirements for the FCNS realisation are given in the following sections. Conformance of the proposed architecture to the defined rules is provided in the FCNS specification, as described in Chapters 4, 5 and 6.

7.2 FCNS Design Process

The FCNS design process aimed at creating a functional prototype of the FCNS stack to verify the logical consistency of its specification and the conformance of the protocol to the rules identified in it. Assumptions and definitions made during the design stages are therefore included in this section, bearing information as to the possible environments in which FCNS would be executed.

The ultimate goal of the FCNS design has been the identification and prevention of errors that could render FCNS operation impossible. Deadlock and livelock avoidance has been of utmost importance for the proposed architecture, since these are situations the protocol may not recover from once realised. Emphasis has also been given to minimising design errors, such as incomplete or contradicting sets of communication rules.

7.2.1 Service Specification

FCNS has been designed to provide a reliable and secure end-to-end transfer service of user and/or signalling information for PS architectures. This service includes the following parameters:

- Connection establishment
- Connection termination
- Transmission error detection, recovery and correction
- Flow control strategy

- Communication channel security
- User data and interlayer security services
- Data encoding
- Full duplex transmission

Depending on the peer requests, this service can either be connection-oriented or connectionless. The former implies the confirmation of the request made by the sending instance, whereas the latter represents an unconfirmed situation in which only an indication is sent to the receiving peer as to the action taken by the transmitter. The assumption made in both cases is that peers are willing to accept the connection request. In any other case, communication cannot proceed resulting in the immediate release of any resources held. FCNS provides acknowledged data transmission facilities as the means of protecting both entities from possible communication attacks, as detailed in Chapters 3 and 5.

Further definitions regarding the FCNS architecture include the possibility of connection termination and/or abortion by either party and the ability of the system to recover from any message losses that may occur. The probability of an undetected error has been chosen to be in the order of 10^{-7} to 10^{-9} for the protocol to maintain an adequate degree of functionality. The implementation of error detection and correction techniques ensures the recovery of FCNS from greater error probabilities.

The architecture must also be able to obtain information concerning security parameters from nodes in the topology. Specific databases and/or elements set by the network operator must be interfaced with the FCNS instance to enable information updates and modifications. Therefore, an external interface mechanism has been installed in the SL to provide for system management services to be offered to the running FCNS instance.

Services have not been designed to provide support for circuit-switched applications. Compatibility with protocols such as the Mobile Application Part and Mobile IP has been preserved only in the sense of carrying the respective messages if that is ever needed. The FCNS is not used to set up and maintain calls to provide telephony and other voice-based services to its users.

However, the FCNS architecture could be used for the transmission of multimedia services, such as real time video and audio. Since its design provides a general secure architectural framework, network operators could use existing protocols to serve as the FCNS communication layers, in an attempt to support a complete network communication solution. Application layer type messages can be mapped into the FCNS messages and be transferred across a packet network topology, irrespective of the data contents. Consequently, an increased flexibility is provided in supporting various connection parameters and situations.

7.2.2 Environment Parameters and Assumptions

FCNS supports full duplex transmission in communication channels, varying from copper cables to optical fibres. Depending on the medium technology used to support the association, a minimum amount of data bits should be calculated to provide a common reference for the FCNS implementations. As an example, the following consideration is given for optical environments [13].

$$velocity = \frac{2}{3} \times c = \frac{2}{3} \times 300,000kms^{-1} = 200,000kms^{-1}$$

For a typical 10km distance it would therefore take $50 \mu s$ for a message to be transferred.

Depending on the transmission speed offered by the supporting hardware, the number of data bits that can be exchanged for the $50 \mu s$ time interval may vary.

Hence, for a modem running at 56Kbps, the amount of bits exchanged in the calculated time interval is *2.8 bits* and for a 2Mbps high-speed modem *100bits*.

The choice of the underlying medium has also been based on security issues, such as the ease in tapping the channel to obtain information as to the traffic in transit or to cause a DoS attack. Optical fibres do not leak light and, since optical transmission does not exhibit electromagnetic emissions, are more difficult to attack at a physical level [177]. Despite this fact, attacks can be successful in optical networks, although the amount of effort and cost that would be required for such a result is significantly greater than that for a shielded coaxial cable. It is consequently recommended that all-optical architectures be adopted in PS environments to enhance the physical security services provided by the FCNS.

Further issues surrounding the environment of the FCNS application include services supporting the requested QoS by the users. Network routing, congestion control and compensation for queuing delays are notions for which respective mechanisms have been added to the FCNS design, including flow control and route calculation procedures.

Additional parameters concerning the types of errors that could occur have been accounted for. Transmission channels have been assumed to impose several effects on the transit data, such as message loss and insertion, message duplication, message re-ordering, error insertion, contents modification and information misrouting. Depending on the fault condition, FCNS has been afforded functions counteracting transmission errors to preserve connection continuation. As far as the types of errors that can occur, this work has assumed a bursty nature, since it provides an adequate approximation of real-network situations. Single-bit and two-bit errors categories are also possible, though the technology in error detection and correction mechanisms (CRC-32) ensures the almost certain recovery of the system from such faults [145].

As an aside on the transmission channel effects, the FCNS provides traffic padding mechanisms and link-based security to account for unauthorised traffic monitoring. An attacker has been assumed to hold the necessary hardware and software tools for launching passive and active network attacks, seeking knowledge of the contents of the data messages or the simple connection disruption.

Transmission speed and bandwidth (BW) limitations depend on the underlying sub-medium and the type of network hardware used to support the topology. Typical examples include 200Mbps shielded twisted pair cables or 200Mbps single mode optical fibres. The identification of the channels has only been made for simulation purposes and hence the observation of the FCNS efficiency in delivering the user data under different network parameters (Chapter 8). As far as security is concerned, no categorisation has taken place on the communications channel, resulting in the same mechanisms used irrespective of the medium.

7.2.3 Protocol Vocabulary

Protocol vocabulary services include the identification of the respective messages used for establishing, maintaining and terminating a connection. Internal and external FCNS messages have been described in a packet format basis, and classified according to the functionality they provide to the FCNS architecture. Messages internal to the FCNS operation are never exchanged between peers and are secured with different secret keys than those used for the user data. Similarly, external FCNS messages cannot be exchanged between the FCNS communication layers and/or the SL.

The peer datagrams, packets and frames hold different headers than internal messages and are only processed by the intended protocol instance. If another layer protocol receives such a message, then it is discarded and an error notification is sent to the SL upon persistence of the condition.

Table 7.1 provides a representation of the messages exchanged between the peer FCNS entities, whilst Table 7.2 illustrates some of the internal messages to the FCNS operation.

Table 7.1: FCNS external messages

Message name	Usage
NODE_CONNECT	Initial connection request to destination (check if peer can accept information)
NODE_REQUEST	Information on the data to be exchanged – connection must have been established
NODE_REQUEST RESPONSE	Response upon the acceptance of the data
NODE_ESTABLISH	Negotiation of presentation/encoding issues
NODE_QOS	QoS parameters request
NODE_QOS_RESPONSE	QoS levels requested
NODE_SYNC	Synchronisation points and parameters
NODE_IDLE	Node instance returns to idle state
COMM_ABORT	Abort the communication
NODE_END	Node has finished all transmissions
NODE_TRANSFER	User data to the UDSES layer and UDPRES peer
MSG_CREATE	User data to the TX_LAYER and UDSES peer
MSG_AGREE	End-to-end QoS support negotiation between TX_LAYER entities
DATAGRAM	User Data to EE_LAYER and TX_LAYER peer
DATAGRAM_SAR	Segmented user data to EE_LAYER and TX_LAYER peer
PACKET	User data to PHYS layer and EE_LAYER peer
PACKET_SAR	Segmented user data to PHYS layer and EE_LAYER peer

FRAME	User data to destination via communications channel (peer PHYS layer entity)
EOF	No more user data to send – wait for response before NODE_END transmission
EOF_RESPONSE	Response to the EOF message bearing its acceptance/rejection
ACK_FRAME	Acknowledgment for the frame or block of frames
NAK_FRAME	Unacknowledgment for the frame or block of frames
ERROR_REP	Error report message of the FCNSEP
ERROR_RESP	Error response message to the ERROR_REP
NAK_REP	Unacknowledge reception of the FCNSEP messages
ROUTE_INFO	Route information for the network nodes

Table 7.2: FCNS internal messages

Message name	Usage
ACK	Acknowledgment of an internal message (if necessary)
NAK	Unacknowledgment (negative acknowledgment) of an internal message
NODE_ALERT	Alert the SL of an error condition (triggers the FCNSEP)
SERVICE_PRIMITIVE	Transmission of the service primitives
NODE_NEI	Verify communication peers
NODE_ACCEPT	Node acceptance
NODE_REJECT	Node rejection
ENCRYPT/DECRYPT	Security context for the internal messages
ENCRYPT_ETE/DECRYPT_ETE	Security context for end-to-end mechanisms
ENCRYPT_LE/DECRYPT_LE	Security context for link-based mechanisms

PATH_ROUTE	Calculated route information list
ACK_ROUTE	Acknowledge chosen route
NAK_ROUTE	Reject chosen route
ADDY_FORMAT	Identify correct address formatting rules – must proceed the PATH_ROUTE
ADDY_ACCEPT	Accept formatting options
ADDY_REJECT	Reject formatting options
MSG_READY	Check PHYS capability in accepting the data messages (from the EE_LAYER)
MSG_STATUS	Identify whether subnetwork can accept and route data (from the TX_LAYER)
TRANSFER_DI	UDSES checks if TX_LAYER can accept data
COMM_QOS	QoS parameters to be supported on an end-to-end basis (UDSES to TX_LAYER)
SET_STATUS	Return to idle state
DECISION	Decision from the source leading to the request primitive
DECISION_RESPONSE	Response from the destination leading to response primitive
PERMISSION	Send permission for sending and/or receiving user data
ACK_DATA	Acknowledge connection parameters requested by the application instance
NAK_DATA	Unacknowledge connection parameters requested by the application instance

Both tables are representative of the messages used in the FCNS realisation. Their identification has been essential in realising the message format and the number of bits that is required to explicitly recognize each one of them.

7.2.4 Message Formats

Message format issues concern the representation of the data transmitted via the FCNS layers in a form acceptable by all communicating parties.

Firstly, the packet encoding issues have been identified to preserve consistency in the FCNS design. The three main possible formatting options have been as follows [6]:

- *Bit oriented formatting.* The data is sent as a stream of bits, with framing flags being used to recognize the adjacent frames.
- *Character oriented formatting.* The data is sent as a stream of frames, with delimiters being used for indicating the beginning and end of each individual frame. These are the Start of Text (STX), End of Text (ETX) and Data Link Escape (DLE) characters (ASCII terminology).
- *Byte-count oriented formatting.* A technique similar to character orientation, but in this case the sender includes in the message the exact number of bytes contained. Since such information is provided, the ETX and DLE characters can be omitted.

Byte-count oriented formatting has been chosen for the FCNS, due to the efficiency it presents in relation to the other two mechanisms.

These procedures aim at finding the minimal message size required to represent the information that is to be sent [6]. The respective calculations took place on the PHYS layer frame, since that is the ultimate message sent between the peer entities. Figure 4.18 of Chapter 4 identifies the frame structure for the FCNS architecture. Since there exist 60 messages in total, then 6 bits are sufficient for indicating the exact message sent, distinguishing also the internal and external messages ($2^5 \leq 60 \leq 2^6$). This number should be added to the frame header

making up the overall frame header size used for calculating the optimum message length for a given error-free interval. Therefore,

- The length of the frame overhead is *17 bytes* or *136 bits* (Figure 4.18)
- The data bits are denoted as $8D$ to result in a byte-multiple payload, D being the payload size in bytes
- The length of the ACK message is *12 bytes* or *96 bits*
- Let p_a and p_d denote the probability of an erroneous ACK and frame respectively
- The total transmitted bits will then be $136 + 96 + 8D = 232 + 8D$ with the total overhead being *232 bits* or *29 bytes*.

1. For an error-free transmission, $p_a = p_d = 0$

Protocol Efficiency (E) will be: $E = \frac{8D}{232 + 8D}$ **Equation 7.1**

Since the FCNS efficiency is independent of the probabilities that any messages will be in error, the data payload size can be as large as possible

2. For an erroneous transmission, $p_a, p_d \neq 0$

The probability that a message and its acknowledgment are error-free is $(1 - p_d)(1 - p_a)$. Hence, the probability that a message is retransmitted becomes $p_r = 1 - (1 - p_d)(1 - p_a)$. Therefore, the probability that it takes i subsequent transmissions to get the message across using $i - 1$ retransmissions and one successful transmission is $p_i = (1 - p_r)p_r^{i-1}$. Then, the expected number of transmissions per message is

$$R = \sum_{i=1}^{\infty} i \times p_i = \sum_{i=1}^{\infty} i \times (1 - p_r)p_r^{i-1} = (1 - p_r) \sum_{i=1}^{\infty} i \times p_r^{i-1} = (1 - p_r) \sum_{j=0}^{\infty} \sum_{i=j}^{\infty} p_r^i$$

$$R = (1 - p_r) \sum_{j=0}^{\infty} \frac{p_r^j}{1 - p_r} = \sum_{j=0}^{\infty} p_r^j = \frac{1}{1 - p_r}$$

Hence, the relative protocol efficiency will become

$$E = \frac{8D}{R(232 + 8D)} = \frac{8D(1 - p_r)}{232 + 8D} = \frac{(1 - p_a)(1 - p_d) * D}{29 + D} \quad \text{Equation 7.2}$$

For a representative error-free interval of 125msec (a Bit Error Probability (BER) of 8×10^{-6}), then:

$$p_a = \frac{96}{125,000} = 7.68 \times 10^{-4}, 1 - p_a = 0.999232$$

$$p_d = \frac{136 + 8D}{125,000}$$

The calculated relative FCNS efficiency is depicted in Figure 7.1. As can be observed from the graph, FCNS achieves a maximum efficiency of 91.57% for a data payload size of 5000 bits or 625 bytes . This means that under erroneous situations with a BER of 8×10^{-6} , the probability that a message is successfully delivered to its destination is 0.9157.

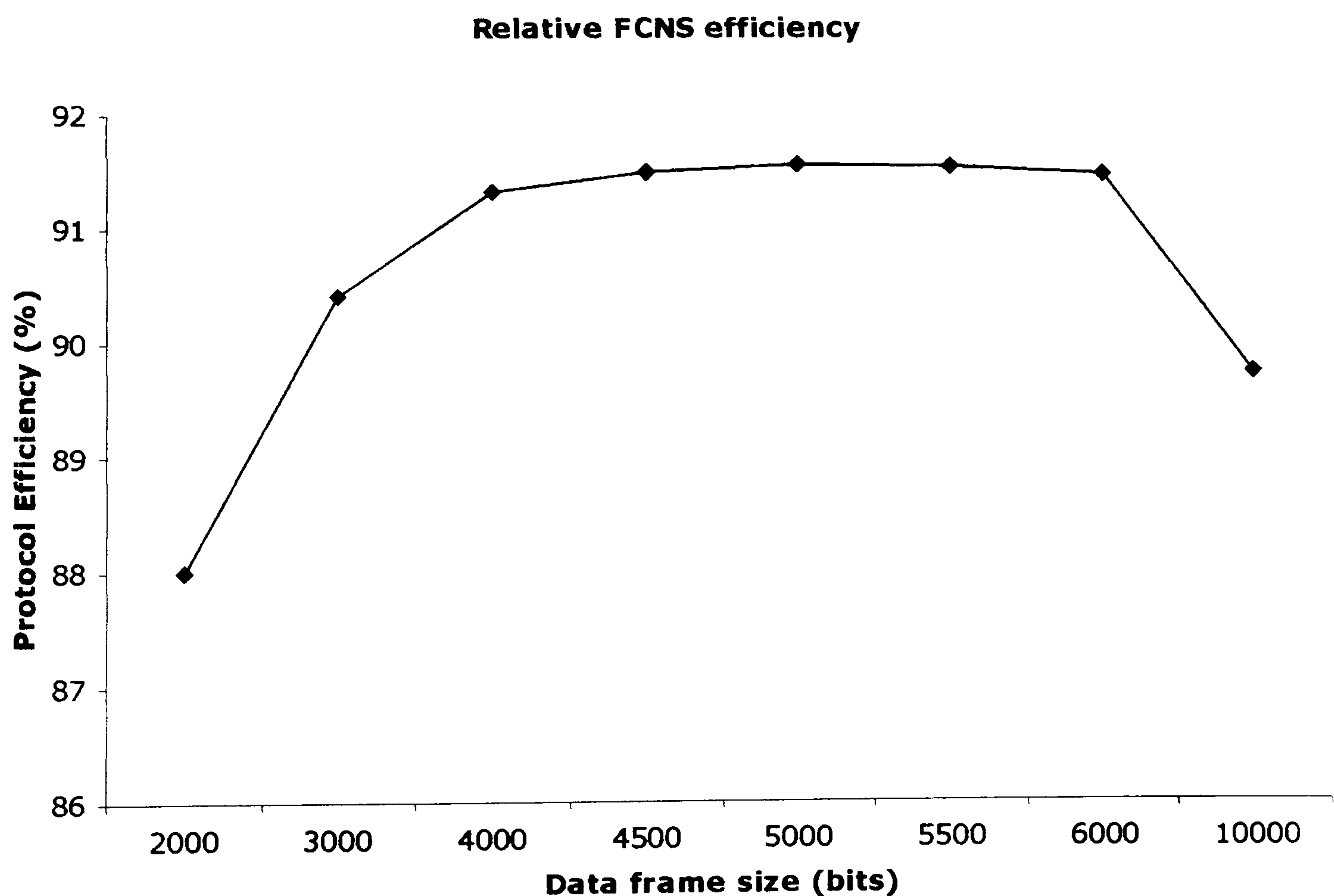


Figure 7.1: Relative FCNS protocol efficiency

The final step in validating the logical consistency of the FCNS specification has been the creation of an abstract high-level prototype. The procedure rules tested in the abstraction enabled the identification of any errors in the FCNS layer interactions and limitations of the protocol in providing the data transfer services.

7.3 FCNS Prototype

Procedure rules are an essential part of the FCNS design, since they encompass the policies and parameters governing the message exchange process. The aim is to devise a complete set of rules, avoiding duplication and ensuring clear definitions of the protocol functions. The approach is to therefore produce a simple design, where overall functionality is broken down to individual pieces (layers), each of which performs a specific task [174], [176]. At the same time, robustness needs to be preserved, to address recovery issues from possible software errors.

7.3.1 Prototype Building

The FCNS prototype has been built using the XSpin validation tool [178] on a Windows 2000 machine. The interest for this work has been the semantics of the protocol and not its precise syntax, creating a high-level abstraction of the FCNS stack. The contents of the messages and transmitted files have been of no importance to the design, hence, a generic model has only been realised, which is depicted in Figure 7.2.

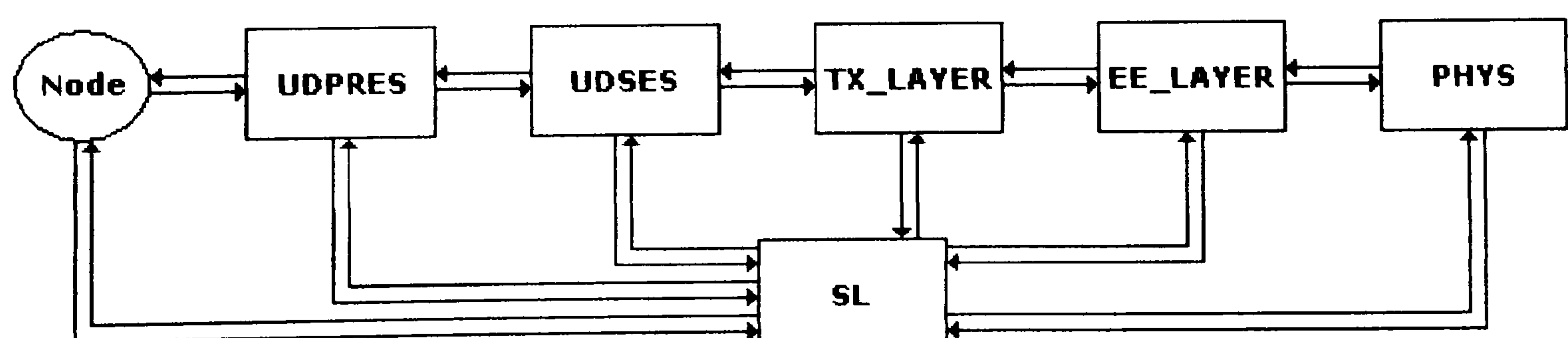


Figure 7.2: FCNS prototype model architecture

The connections between the layers of the prototype are channels forwarding information from and to a layer, without introducing any delays or bit errors. The omission of an application layer from the validation model follows the principles described in Chapter 4. The inclusion of models implicit to the validator for the sending and receiving instances has only been done for the provision of a continuous data generator and sink, in the form of the *node* functional block. Similarly, a physical layer defining transmission characteristics and capabilities falls outside the scope of the FCNS prototype.

The model of Figure 7.2 lacks also the provision of distinctive input and output FCNS abstractions. The reason for this approach is that the prototype validation has solely focused on the FCNS layer interactions and not the communication process between peers. Since every network node should be able to both send and receive data, the FCNS blocks of the prototype architecture entail the necessary functionality to compensate for both situations.

The definitions of the prototype parameters have been based on the PROMELA language of the XSpin software package [6]. Further information on formal description languages used for prototype building is described in [179 - 181]. For the creation of the prototype, the following functions have been assigned to the FCNS communication layers and the SL:

- UDPRES block: Interface to the node process handling re-transmission requests. Transfer syntax negotiation is only checked as a message exchange process and not in terms of message encoding. It is also the block responsible for initiating and terminating a connection depending on the user requests.
- UDSES block: Controls the data transfer process and provides any necessary synchronisation parameters. Maintenance of the data to be transferred is also accounted for.

- TX_LAYER block: Implements a flow control mechanism for the end-to-end transmission. It handles the sending and receiving of the data as well as its acknowledgment depending on the window size.
- EE_LAYER block: Verifies with the SL the path in which messages will traverse and calculates a checksum value appended in the packet for error detection purposes.
- PHYS block: Forwards the data to the physical link, implementing a sliding window ARQ mechanism for handling re-transmissions.
- SL block: Provision of end-to-end and link-based security mechanisms. It checks the integrity of the messages exchanged between the FCNS communication layers and supports the implementation of the traffic padding mechanism.

7.3.2 Validation Tool Applicability

XSpin [178] is a high-level design verification tool, used for models of applications and not implementations. The procedures surrounding the operation of XSpin are based on abstract models, for which details must be suppressed to create a prototype based on only the minimum necessary functionality.

The tool provides three types of simulation:

1. Random simulation option used to debug a model. No correctness requirements are checked during simulation runs. All non-deterministic decisions are resolved randomly.
2. Interactive simulation can be used to force the execution towards a known point. The user is prompted at every point in the execution where a non-deterministic choice has to be resolved.
3. A guided simulation is used to follow an error-trail that was produced by the verifier for every FCNS functional block.

The advantage of the simulation procedures is the production of time-sequence diagrams, where the verifier outlines processes in terms of states created during simulation time. States and layer interactions can be checked via these diagrams, providing the designer with an indication as to the correctness of the model and its compliance to the specification.

Further advantages include the ability to analyse the properties of the system interactions, clearly identifying the internal states of a communication protocol [182]. It also enables the location of design errors in the abstract model representation, offering an early protection measure in recognizing potential implementation pitfalls.

The limitations of XSpin lay in the usage of excessive CPU and memory loads for the verification processes. Representative values obtained by the FCNS prototype are depicted in Table 7.3.

Table 7.3: Memory requirements of the XSpin verification tool for the FCNS model

FCNS layer	Memory requirement (MB)
UDPRES	35.759
UDSES	35.759
TX_LAYER	35.759
EE_LAYER	134.623
PHYS	15.112
SL	214.223

The amounts provided in Table 7.3 are quite large compared to the level of abstraction the FCNS communication layers have undergone. Further interactions and processes have increased the memory requirements of the validation tool, limiting its usage for this thesis.

Additionally, the provision of verification procedures only for the layer interactions meant that the processes and FCNS functionality could only be tested on the complete design. Software code flaws and miscalculations have decreased the amount of time it took for the FCNS software realisation to take place, and the testing of a more complex system than the prototype at the same time as the design process was progressing.

Finally, a weakness this work has identified has been the lack of automated implementation procedures, whereby software code could be produced in a high-level language such as C/C++. Such a feature would decrease the time it took to complete the FCNS implementation, since after the verification of the desired model properties the task would focus on only the addition of the functionality left out of the prototype. Instead, once the FCNS model had been verified, the work had to move into manually realising the FCNS layers into software.

7.4 Source Code Implementation

Following the verification of the FCNS specification logical consistency and the conformance of the model to the requirements set in the design, the OMNET++ simulator [11] has been used to realise the FCNS model. The idea behind the implementation of the protocol into a software entity has been the provision of a formal test-bed in observing issues related to the FCNS performance and efficiency in delivering the requested services.

7.4.1 FCNS Stack Simulation Environment

The provision of the FCNS stack simulation environment enabled the identification of the performance of the protocol in supporting the services designed in the specification. Message forwarding, address verification, security context exchanges and route calculations have been the major functions tested. Following the

validation of the prototype, where layer interactions have been verified, the model depicted in Figure 7.3 has been used to test the processes between peer entities.

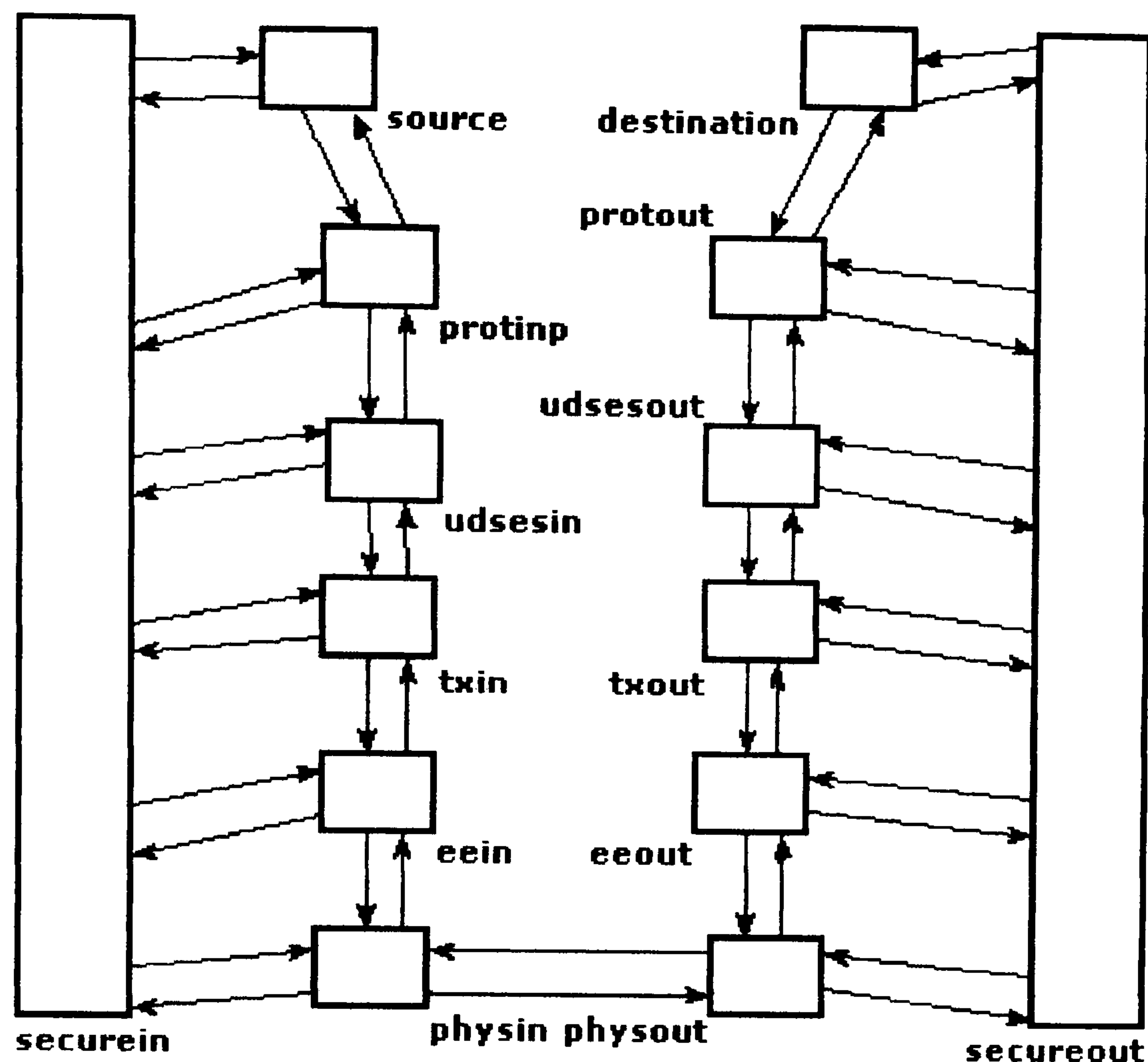


Figure 7.3: FCNS stack simulation environment

The specific model aimed at testing the capabilities of the FCNS in sending and receiving information that might be required in a PS environment. The *Source* and *Destination* module instances represent a data generator and a sink respectively, continuously producing and receiving data for the communications channel. The *Protinp* module represents the functions of the UDPRES layer at the sending instance, whilst *Udsesin* the functionality of the UDSSES layer. *Txin* encompasses the details of the TX_LAYER, *Eein* the ones provided by the EE_LAYER, whereas *Physin* and *Securein* those for the PHYS and the SL layers respectively. Similarly, the *Protout*, *Udsesout*, *Txout*, *Eeout*, *Physout* and *Secureout* represent the output instance of the FCNS stack.

The links between the FCNS layer instances did not introduce errors but delays representing the amount of time it would be required to process a message.

Typical values ranged from hundred microseconds to a few milliseconds depending on the message size and the encryption technique applied.

Additionally, the link between the PHYS instances has been regarded as a communications channel, with the characteristics outlined in Section 7.2.2. Measurements have consequently been taken with respect to this link, including protocol efficiency calculations and throughput measurements for various network situations.

The functions of the OMNET++ simulator enabled the setting of the respective parameters in changing delay and BER values, with the results obtained being presented in Chapter 8.

7.4.2 Simulation Tool Applicability

OMNET++ is a discrete event simulator, enabling protocol procedures to be programmed and realised using the functionality of the software tool. Functions such as message encapsulation and decapsulation, as well as route calculations have been based on those provided by OMNET++, simplifying the FCNS implementation process. The tool offers the flexibility of explicitly defining parameters and variables for the environment, representing the layers of the protocol in individual modules, such as those depicted in Figure 7.3. The designer is able to address specification issues irrespective of the platform the protocol will run, removing dependencies from existing protocol architectures such as the IP and TCP.

The tool lacks standardised module libraries, enhancing the difficulty of the FCNS realisation and the creation of case studies for measuring the protocol efficiency in real-network type situations. Software tools such as OPNET [183] provide the capabilities of model platforms, including wireless networks and the UMTS CN, but

licences are very expensive. Similar functionality had to be programmed in the OMNET++ simulator, reducing the flexibility of the design. At the same time, the amount of effort that has been required to provide a complete test-bed architecture has been increased, at the expense of a controlled environment with parameters clearly defined by the designer.

Furthermore, the work has identified the inability of OMNET++ to handle message structures inside queues. The use of pointers has been acknowledged to remove the ability of the simulator to distinguish between the messages in a buffer [184]. Consequently, whenever there has been the need of simulating a message buffer and/or a queue, the message C++ structures have been replaced by the *addPar()* simulator function, manually adding the appropriate parameters to the FCNS messages. Although this did not have any effect on the continuation of the simulation runs, it leveraged the amount of effort and time taken to provide a complete solution.

Moreover, OMNET++ had limitations in interpreting the implicit message structures into byte-type buffers. Interfacing the FCNS design with cryptographic libraries such as the Cryptlib [185] has therefore become impossible, leading to simulating only the effects security had on the FCNS communication.

The nature of the simulation tool meant the representation of the FCNS architecture using discrete techniques. The implications of this approach were that protocol states could only change at particular time intervals and that all variables maintained their values throughout these time periods. This meant that processes could not occur simultaneously but by following a specific order. In the FCNS simulation context this implied the loss of simulation time in processing the data messages sent, since the input instance has been unable to send information prior to the interpretation of the previous message by the destination. As a result of this, continuous data generation entailed either the provision of a message buffer at

the input PHYS layer instance to store the respective messages, or the halting of the sending process for an amount of time equal to that taken for acknowledging a message.

Table 7.4 depicts a representative case to further address this OMNET++ pitfall. In the example presented, the run has assumed a *5 msec* FCNS message processing delay in delivering the data from the source node to the PHYS layer, with a round trip propagation delay (RTT) of *60 msec*. The overall delay in processing a frame block represents the time it takes for that block to reach the receiving entity plus the time taken to acknowledge its reception. For consistency, a complete frame block has assumed to contain 32 messages (frames).

Table 7.4: Message sending processing time

Sending Rate	Processing time	Overall delay in processing the frame block
32 messages before ACK	160 msec	1320 msec
1 message before ACK	5 msec	2560 msec

The amounts illustrated in Table 7.4 have been calculated as follows:

- For an acknowledgment after one complete frame block (Go-Back N or Selective Repeat ARQ), the processing time for the 32 data messages in one FCNS instance is *160 msec*. For both the input and output instances this becomes *320 msec*. The time to transmit each message is half the RTT and hence for 32 messages this becomes *960 msec*. The acknowledgement should take *30 msec* (RTT/2) to reach the sending PHYS instance plus *10 msec* for its processing, resulting in an overall *1320 msec* time before the sender can transmit the next block.
- For acknowledging every single frame block segment (Stop and Wait ARQ), the processing time for a single message in both FCNS entities is *10 msec*

plus the *10 msec* required for the ACK to reach the sender, making up a total of *20 msec*. Transmitting and acknowledging the message would also take *60 msec* entailing a total of *80 msec* overall delay in processing a single frame and hence *2560 msec* for the frame block.

If the transmission and message reception could occur concurrently, then assuming a Go-Back N ARQ scheme, the overall processing delay would be:

- Total processing time for a frame block in the FCNS instances would be *160 msec*. This is due to the fact that sending and receiving are simultaneous processes. Sending the frame block down the communications channel would take *960 msec* plus *40 msec* for its acknowledgment including the processing time for the ACK message, making a total propagation delay of *1000 msec*. Therefore, the next block could be transmitted at *1160 msec* after the previous one assuming that data transfer continuation should be based on the ACK reception. If the sender is permitted to send without receiving an ACK, then the overall processing delay is further reduced to *1090 msec* (*1160 msec* minus *40 msec* ACK total processing time minus *30 msec* that would take for the last frame to reach the receiving PHYS layer entity).

Finally, the memory consumption of the OMNET++ simulator for each environment module has been observed to be quite large. On a Windows 2000 machine with 256 MB of RAM, the simulation environment that has successfully been built encompassed only 6 nodes, as depicted in the generic model of Chapter 8. Further node additions resulted in the program to crash due to lack of memory. Table 7.5 illustrates typical stack usage memory amounts for the model of Figure 7.4.

Table 7.5: FCNS simulation environment memory consumption

FCNS layer	Memory consumption (input/output instances)
Source / Destination	59032 / 55648 (total of 114,680 bytes)
UDPRES layer	3670324 / 3950820 (total of 7,621,144 bytes)
UDSES layer	24064 / 24816 (total of 48,880 bytes)
TX_LAYER	25004 / 25380 (total of 50,384 bytes)
EE_LAYER	25756 / 23688 (total of 49,444 bytes)
PHYS layer	124644 / 73508 (total of 198,152 bytes)
Security layer	99640 / 63168 (total of 162,808 bytes)

For a single simulation run, Table 7.5 depicts the partitioning of a total of *8,245,492 bytes* of memory consumption for just the FCNS simulation environment. Additions in FCNS functionality and processes would increase this requirement resulting in constraints as to the applicability of the simulator in implementing the FCNS into a large-scale network environment.

Overall, the OMNET++ simulator provided the means of identifying design errors in the FCNS specification, refining the protocol and removing any unwanted states and functions that could lead to potential pitfalls. As a possible further step, the FCNS implementation could be ported to the OPNET environment, further testing the protocol under standardised architectures for wireless telecommunication systems.

Chapter 8: Case Studies

Chapter 8. Case Studies

8.1 Overview

8.2 FCNS Stack Implementation

8.2.1 Simulation Environment Description

8.2.2 Results and Measurements

8.3 Packet-Switched Architecture

8.3.1 Simulation Environment Description

8.3.2 Results and Measurements

In this chapter, observations and results taken from the FCNS simulation environments are presented. The measurements are categorised into two sections. In the first, the FCNS architectural simulation environment is given, depicting the model used to provide a test-bed for the FCNS stack. Results and observations concern the efficiency and throughput of the protocol under situations with bit errors, delays and delay variations (jitter). The second section involves measurements taken by implementing the protocol into a general packet-switched topology to establish the means of measuring the FCNS performance in delivering user data across various network elements. Comparisons are also presented, which have been made against the TCP/IP suite running on a typical 10Mbps Ethernet network.

8.1 Overview

The FCNS specification described in Chapters 4,5 and 6 represents an alternative to the OSI security model architecture, to address the security problems given in Chapter 3. The software implementation of the FCNS aims to provide a measure of the protocol functionality under typical operational conditions.

The specification of the proposal developed by this work cannot be fully compared to existing communication solutions such as the TCP/IP. These have been developed over many years and run on diverse operating systems (Chapter 2). Measuring the performance of a real-network FCNS instance against the Internet suite would require protocol standardisation and operating system implementations that have fallen outside the scope of this work. Consequently, the FCNS simulation environments have been created to enable verification of the consistency of the specification and to give an idea of the FCNS capabilities in comparison to the TCP/IP suite.

In the observations presented in the following sections, comparisons with theoretically calculated values are given, to enable the referencing of the simulated FCNS model against its specification.

8.2 FCNS Stack Implementation

The aim of the FCNS stack simulation environment was the identification and recording of the protocol functions and variables necessary to setup and maintain a packet data connection. The model has been divided into an input and output instance to enable the classification and management of their respective functions. Similarities have been preserved whenever needed, since in a real-network topology the two instances would compose a single protocol occurrence. Therefore, functionality duplication has become inevitable to preserve an adequate level of operation for the simulated model.

8.2.1 Simulation Environment Description

The architecture used to provide the measurements is depicted in Figure 8.1, taken by the graphical interface of the OMNET++ simulator (GNED tool).

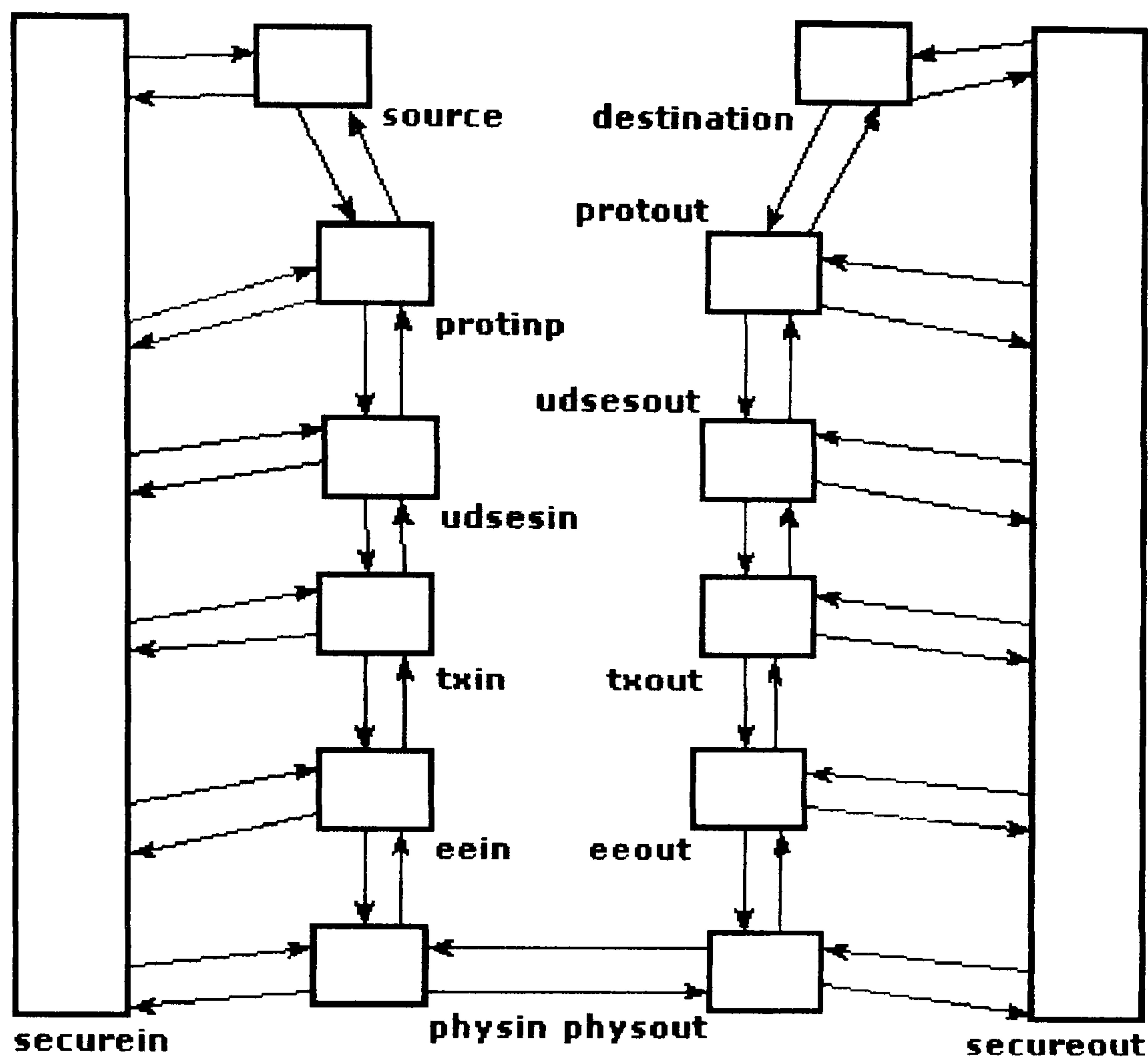


Figure 8.1: FCNS OMNET++ simulation topology

The identification of the individual modules is given in Chapter 7, together with information regarding the links present in the model. For the measurements illustrated in this chapter, the channels between the individual modules have been programmed to introduce small delays to simulate the amount of time it would take for a protocol to process a message.

Table 8.1 illustrates the FCNS communication layer header sizes used throughout the simulation measurements, whilst Tables 8.2 and 8.3 depict those used for the TCP/IP suite.

Table 8.1: FCNS communication layer header sizes

FCNS layer	Header Size (security)	Header Size (no security)
UDPRES	10 bytes	7 bytes
UDSES	15 bytes	12 bytes
TX_LAYER	24 bytes	21 bytes
EE_LAYER	20 bytes	17 bytes
PHYS	17 bytes	14 bytes
<i>Total</i>	<i>86 bytes</i>	<i>71 bytes</i>

The ACK of the FCNS structure for the data messages had a constant overhead of *12 bytes* or *96 bits* when full security measures were applied and *9 bytes* or *72 bits* when no security mechanisms were enforced.

Table 8.2: TCP/IPv4 protocol suite header sizes

Internet layer	Header Size (bytes)	ACK size (bytes)
Ethernet preamble	8	8
Ethernet source address	6	6
Ethernet destination address	6	6
Ethernet type field	2	2
Ethernet CRC	4	4
IPv4 header	20	20
TCP header	20	20
Pad (to Ethernet minimum)	0	6
Inter-packet gap (9.6 μ sec)	12	12
<i>Total</i>	<i>78</i>	<i>84</i>

Table 8.3: TCP/IPv6 protocol suite header sizes

Internet layer	Header Size (bytes)	ACK size (bytes)
Ethernet preamble	8	8
Ethernet source address	6	6
Ethernet destination address	6	6
Ethernet type field	2	2
Ethernet CRC	4	4
IPv6 header	40	40
ESP header	14	14
TCP header	20	20
Pad (to Ethernet minimum)	0	6
Inter-packet gap (9.6 μ sec)	12	12
<i>Total</i>	<i>112</i>	<i>118</i>

The choice of an additional security extension header for the Internet suite has been made to enable the comparison of the FCNS against a secured Internet connection. The ESP was chosen since it provides both authentication and integrity features when running in tunnel mode (Chapter 3).

The values presented in Tables 8.1, 8.2 and 8.3 have been used to measure the individual protocol stack efficiencies at a theoretical level, using equations 7.1 and 7.2 of Chapter 7. This enabled the identification of the maximum possible throughput and stack efficiency, providing the necessary reference point for the simulation results.

8.2.2 Results and Measurements

This section presents the results of the simulation runs categorised according to the variable kept constant, namely either the Bit Error Rate (BER) or the Round Trip Time (RTT) delay.

All measurements have been taken with respect to the following parameters.

- *Maximum number of frames/second (MFS)*: Identifies the maximum obtainable throughput for an errorless transmission. Defined as the ratio of the data rate over the physical frame size. For the FCNS architecture, two possibilities have been identified:
 1. *FCNS with security measures*. With a typical 1460-byte data size [14], then the overall FCNS frame will have a size of 1546 bytes or 12368 bits. This means that for a communications channel of 10 Mbps BW, the maximum number of frames that can be transferred (*MFS*) will be approximately 808 per second.
 2. *FCNS no security*. For the same 1460-byte data size, the overall FCNS frame size becomes 1531 bytes or 12248 bits. Consequently, *MFS* will be approximately 816 frames/second.
- *EE_LAYER Packet Throughput*: Represents the number of payload bits that can be transferred per simulation time second, given the *MFS* obtained.
 1. *FCNS with security measures*. With the *MFS* as above, the size of the frame payload (*EE_LAYER* packet) is 1529 bytes long. The packet throughput (T_{EE_LAYER}) is the product of these, which is approximately 9.98 Mbps. Therefore the throughput efficiency is 98.9% for the 10Mbps data rate.
 2. *FCNS no security*. Similarly, in this case, the frame payload is 1517 bytes giving an efficiency of 99.08%.
- *Time to transmit a single frame*. This has been used to test the response of the FCNS simulation environment in delivering information according to the

theoretical standards. It may be calculated as the number of bits present in the frame over the channel capacity data rate.

1. *FCNS with security measures.* The time to transmit a single frame

$$(T_{1F}) \text{ is: } T_{1F} = \frac{12368 \text{ bits}}{10 \times 10^6 \text{ bps}} = 1.24 \text{ m sec.}$$

This value has been used as

half the RTT time for a set of simulation runs to enable identification of the FCNS software model in delivering user information at the maximum theoretical rate.

2. *FCNS no security.* In this case, the time to transmit one frame

$$\text{becomes } T_{1F} = \frac{12248 \text{ bits}}{10 \times 10^6 \text{ bps}} = 1.22 \text{ m sec}$$

- *Channel Utilisation (CU).* The parameter represents how efficiently the communications channel is utilised by the data transfer process. It can be regarded as the ratio of the maximum achieved throughput over the channel capacity. Problems though can arise if all nodes in a network transmit at the highest possible rate, resulting in a maximum CU. Cases include congestion built up, which has the effect of minimising the throughput of an instance and/or lead to potential message losses, as depicted in Figure 8.2 [28].

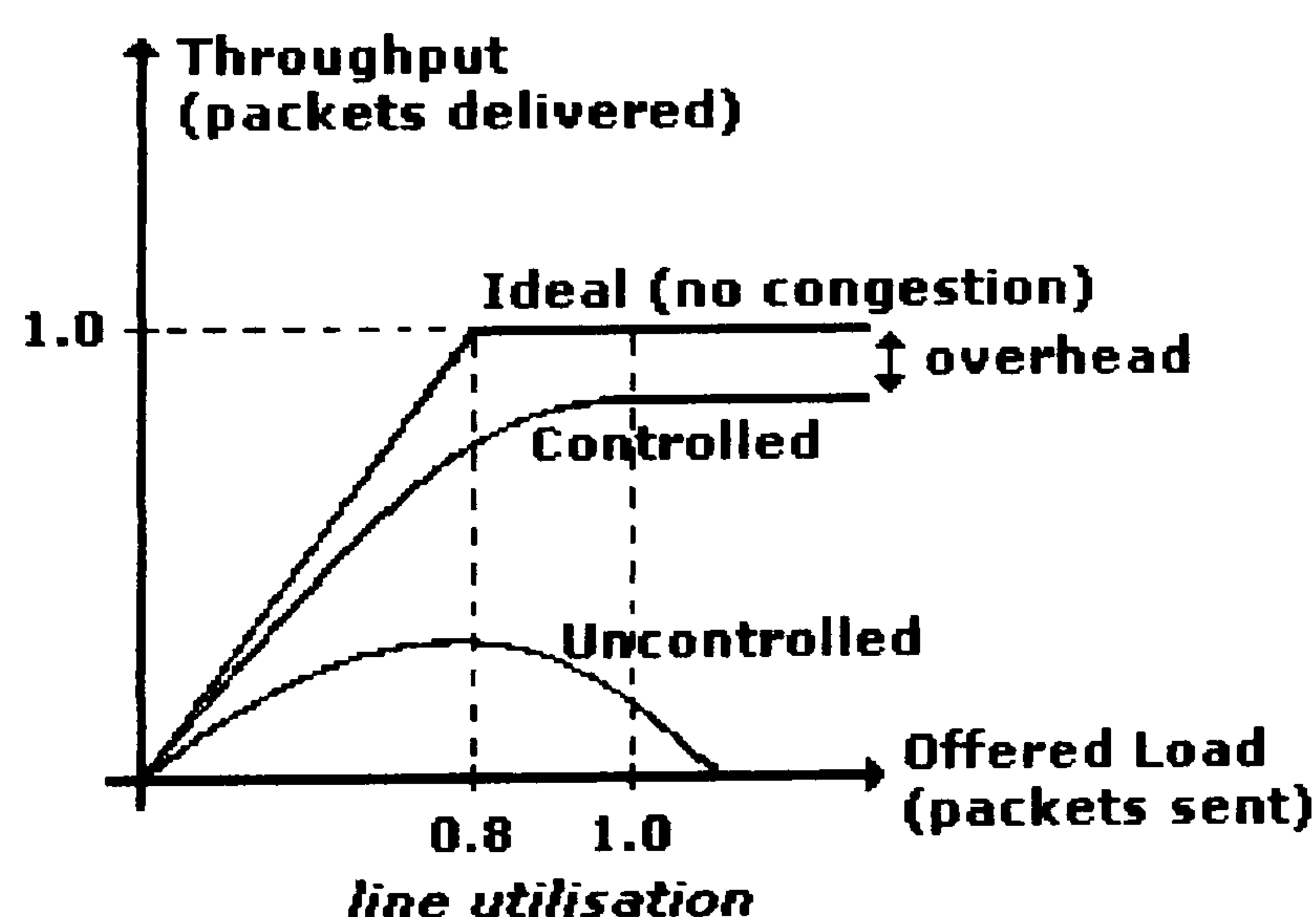


Figure 8.2: Congestion effects on packet throughput

For the uncontrolled case illustrated in Figure 8.2, when the line utilisation exceeds 80%, the packet throughput drops dramatically, reaching a zero value for nodes flooding the lines with the user data. As a consequence,

there has to be a measure of the maximum number of messages in transit, a notion expressed in terms of the bandwidth-delay product.

- *BW-delay product.* Represents the number of bits that can be in transit at any given time, to enable the network achieve the highest possible throughput before network becomes congested. The product can be used in conjunction with the flow control mechanisms offered by the protocol to enable the identification of the number of messages that can successfully be sent before an acknowledgement is requested, given the respective channel capacity. In a sense, this variable determines the receiver window size in terms of the network latency (RTT). As a general reference, links should never be over-utilised, with the nodes achieving a CU of less than 50% [14].
- *Loss and deliverability Ratio.* Loss ratio represents the amount of messages that have been lost either due to the presence of bit errors or due to discarding by the receiver as duplicates and/or intended for another recipient. It is expressed in percentage form as:

$$LossRatio = \frac{Frames_{sent} - Frames_{received}}{Frames_{sent}} \times 100\% .$$

In a similar manner, the *deliverability ratio* $\frac{Frames_{received}}{Frames_{sent}}$ can express the throughput of the particular simulation run.

- *Packet Throughput.* This value is used to depict the efficiency of the FCNS in transmitting the user data under erroneous conditions. It can be expressed in terms of the deliverability ratio or by calculating the overall number of bits sent in a simulation run.

The variables presented above have been used to illustrate the FCNS efficiency compared to the theoretically calculated values and those obtained for the Internet suite.

The throughput response of a general-purpose protocol scheme has also been calculated, representing the reference point for the observations made in the FCNS simulation runs. The calculations follow the assumption of an error probability bearing the characteristics of a Poisson distribution. This hypothesis has been made to enable the representation of the BER probability as a random occurrence in a specified period of time (the simulation run time), matching the characteristics of a real-network environment. The number of bits in the FCNS frame overhead h is *688 bits* when full security measures are applied and *568 bits* with no security functionality. The data size values x had a range from 1600 to 160,000 bits, and the BER from 0 to 0.001. For a Poisson distribution, the probability of incident for a given variable is: $P(x) = \frac{\lambda^\kappa}{\kappa!} e^{-\lambda}$, where λ represents the average number of occurrences and κ the number of actual incidences. For an error rate p , then the throughput becomes $T = (1 - \frac{h}{x})e^{-px} \cong (1 - \frac{h}{x})(1 - px)$, where e^{-px} represents the distribution for a single occurrence, that is, the $P(0)$ [6]. Figure 8.3 corresponds to the throughput obtained for the case of the FCNS with full security measures.

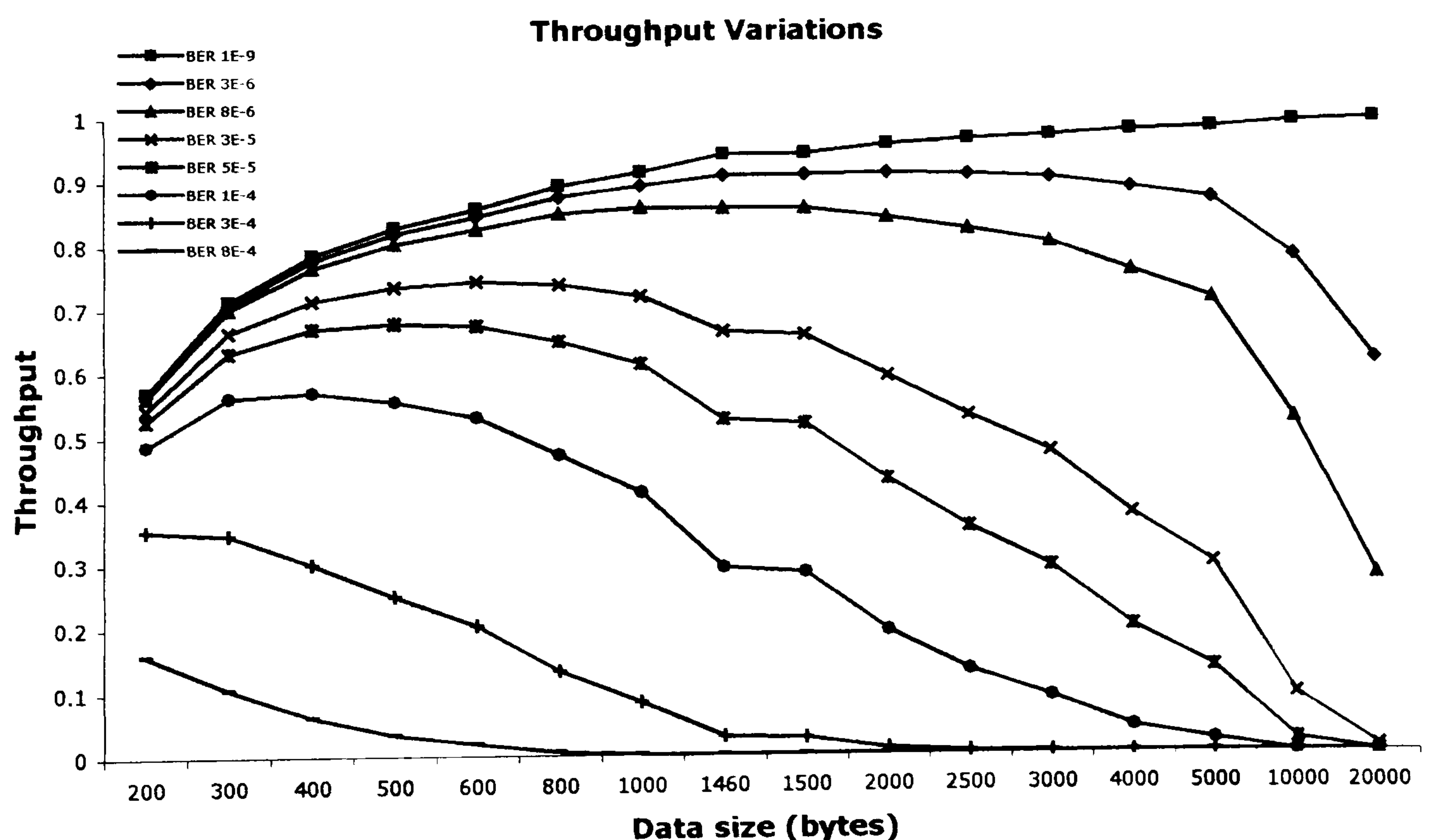


Figure 8.3: Throughput variations – FCNS full security measures

The following sections illustrate the results obtained by the FCNS stack simulation environment with respect to the theoretical calculations presented above.

8.2.2.1 Constant Delay, Variable BER, Reference Measurements

Varying the BER imposed on the FCNS messages under a particular RTT delay resulted in the observations depicted in the following figures.

8.2.2.1.1 RTT of 2.5 msec – Loss Ratio (no timers)

This measurement has been used to identify the potential frame loss that could occur when the network nodes were transmitting at the highest possible rate ($T_{1F}=1.24$ msec) resulting in the maximum MFS. The value of 2.5 msec is very close to the 3 msec RTT of an Ethernet 10 Mbps LAN [14] and can therefore provide a good view of the losses that may incur in such physical network topologies.

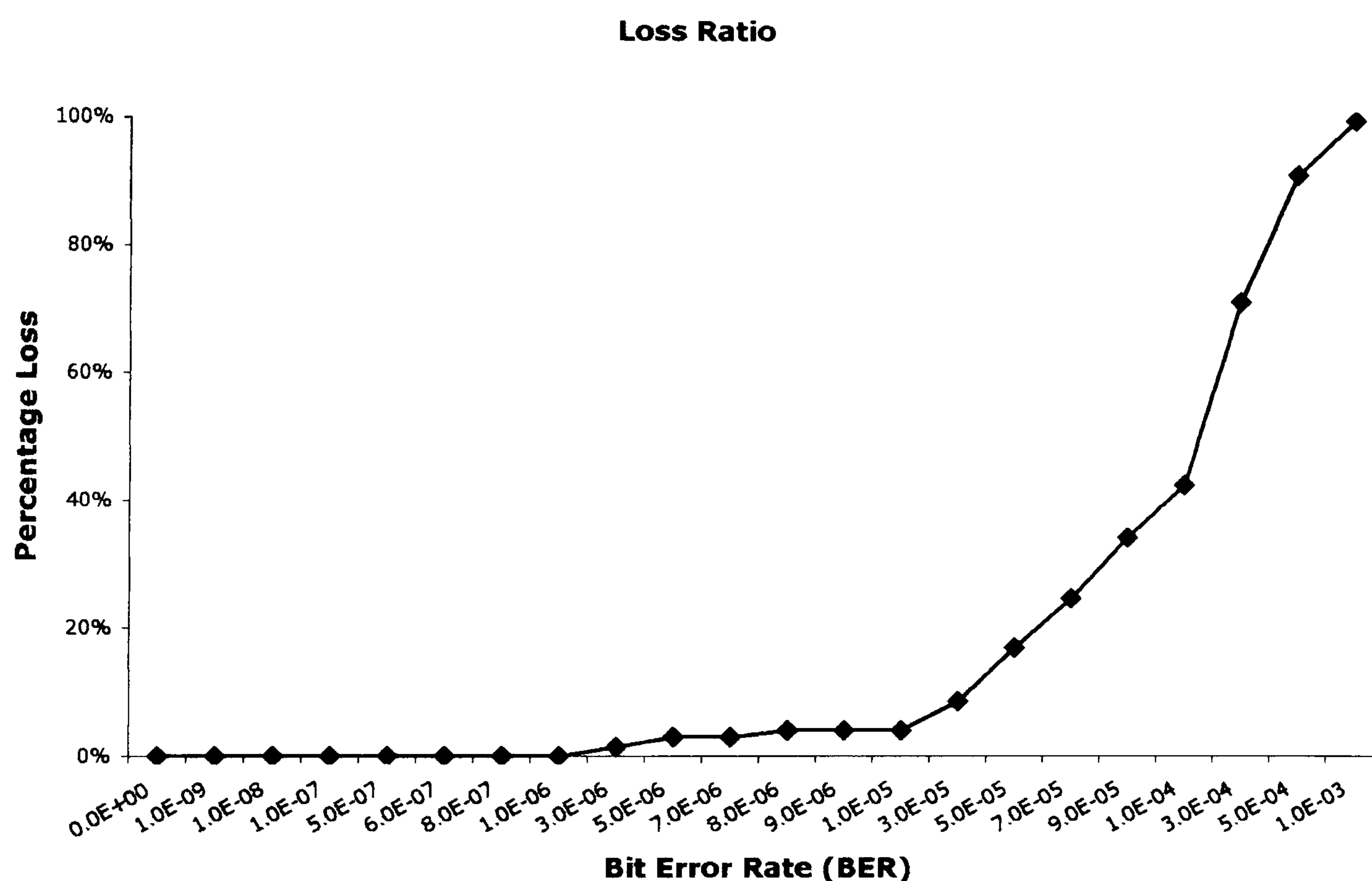


Figure 8.4: Reference loss ratio for RTT of 2.5 msec and variable BER

This RTT value has also been used as a reference point for the proceeding observations to enable the identification of the FCNS stack response in delay and BER variations. For these particular sets of measurements, all FCNS layer message

timers have been disabled, to achieve the maximum possible throughput at a theoretical level.

8.2.2.1.2 RTT of 2.5 msec – EE_LAYER Packet Throughput (no timers)

The FCNS packet throughput observation of Figure 8.5 represents the amount of EE_LAYER packets that have been sent in the reference simulation set, for sending rates matching the optimum T_{1F} . The variations indicated in the measurement are due to the simulator memory usage resulting in different number of packets being sent for the various BER cases.

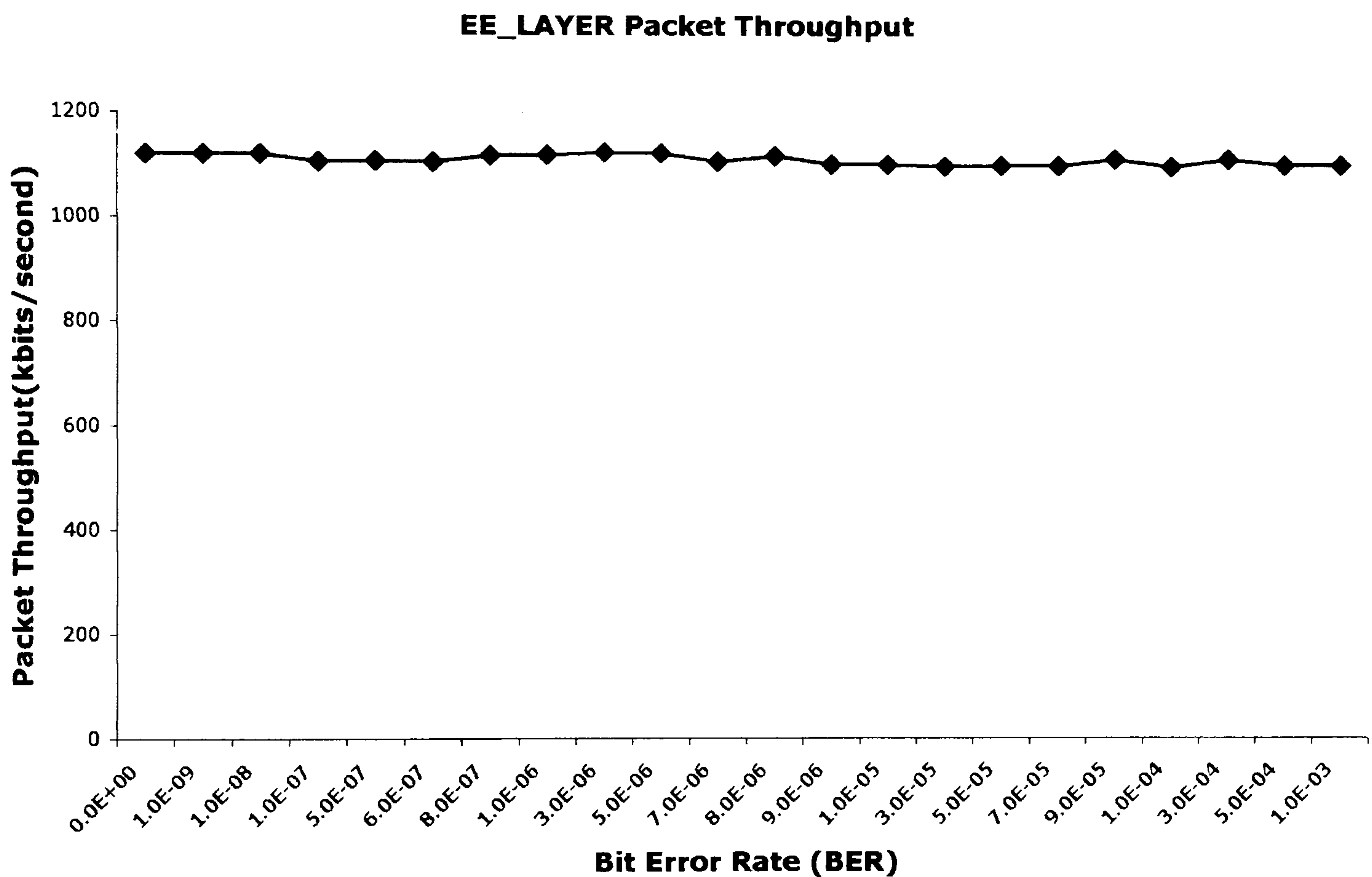


Figure 8.5: Reference FCNS packet throughput for RTT of 2.5 msec and variable BER

The jitter observed falls into the area of 250 kilobits/second or 31,250 bytes, that is, 2.5 packets/second. Given the processing load alterations and the applicability issues of the OMNET++ simulator described in Chapter 7, the packet variation rate observed can be regarded as acceptable.

The maximum obtainable packet throughput is 1,124,412 bits, which results in a CU of 11.24% for the whole of the simulation time period. For a single node

transmitting at the highest MFS, flow control issues may be omitted, since the communication links may not built up congestion. In contrast, for a large-scale network topology, the amount of bits that can be sent by the nodes should be limited to the BW-delay product of 25,000, to provide for a delay-free data transfer process.

8.2.2.1.3 RTT of 2.5 msec – Frame Throughput (no timers)

The final observation made for the ideal FCNS simulation model concerns the overall frame throughput variation for the duration of the simulation runs. In the FCNS measurements, the work has assumed a constant data size throughout the set of the runs. Data length alterations have taken place to provide the means of measuring the protocol's performance in delivering different kinds of information including network-signalling data. Such results are provided in the following sections, where comparisons are made against theoretical calculations.

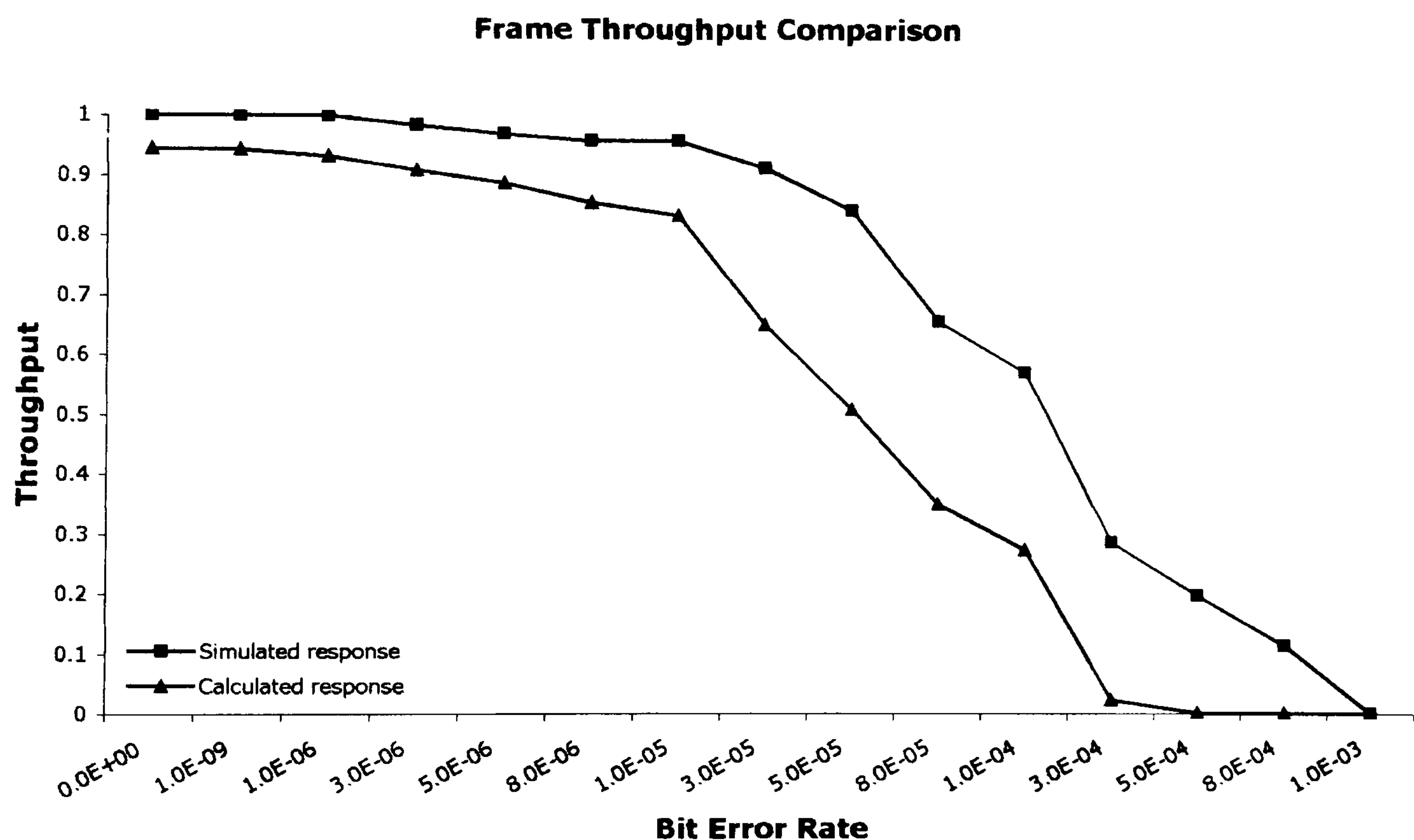


Figure 8.6: Throughput comparison – reference RTT 2.5 msec and theoretical calculation

It may be seen that the simulated frame throughput is greater than the theoretical (*Calculated response* caption) value (Figure 8.3). This is due to the lack of error detection and correction mechanisms in the theoretical calculations. The presence

of such schemes in the FCNS stack architecture results in erroneous messages being corrected and consequently forwarded to the upper communication layers. Lack of such techniques increases the loss ratio and therefore minimises the deliverability of the FCNS frames.

With this provision, it may be concluded that the representation illustrated in Figure 8.6 can be an acceptable measure of the FCNS performance acting as a reference (*Reference observation* caption) for the remainder of the observations.

8.2.2.2 Constant Delay, Variable BER

For these sets of simulation runs, the respective timers at the source and destination FCNS instances have been enabled, to create a more realistic view of the FCNS stack topology. The most important timers concern the acknowledgement and message reception values, which may vary depending on the network FCNS runs on. Typical values range from 50 msec to 260 msec to support a wide range of observations to take place.

8.2.2.2.1 RTT of 2.5 msec – Loss Ratio

In the results presented in Figure 8.7, a comparison is provided with the reference observations of Section 8.2.2.1.1.

The response of the FCNS protocol stack to message losses is almost identical, as presented in Figure 8.7. This comes as a result of the timers' rates for the FCNS instances, which have a minimum value of 50 msec to preserve a level of consistency for the simulation runs. Any period below the 50 msec threshold is possible (25 msec is the minimum acceptable), but its exclusion has been necessary to simulate a realistic model able to adapt in topologies with a large number of intermediate nodes between the peer entities.

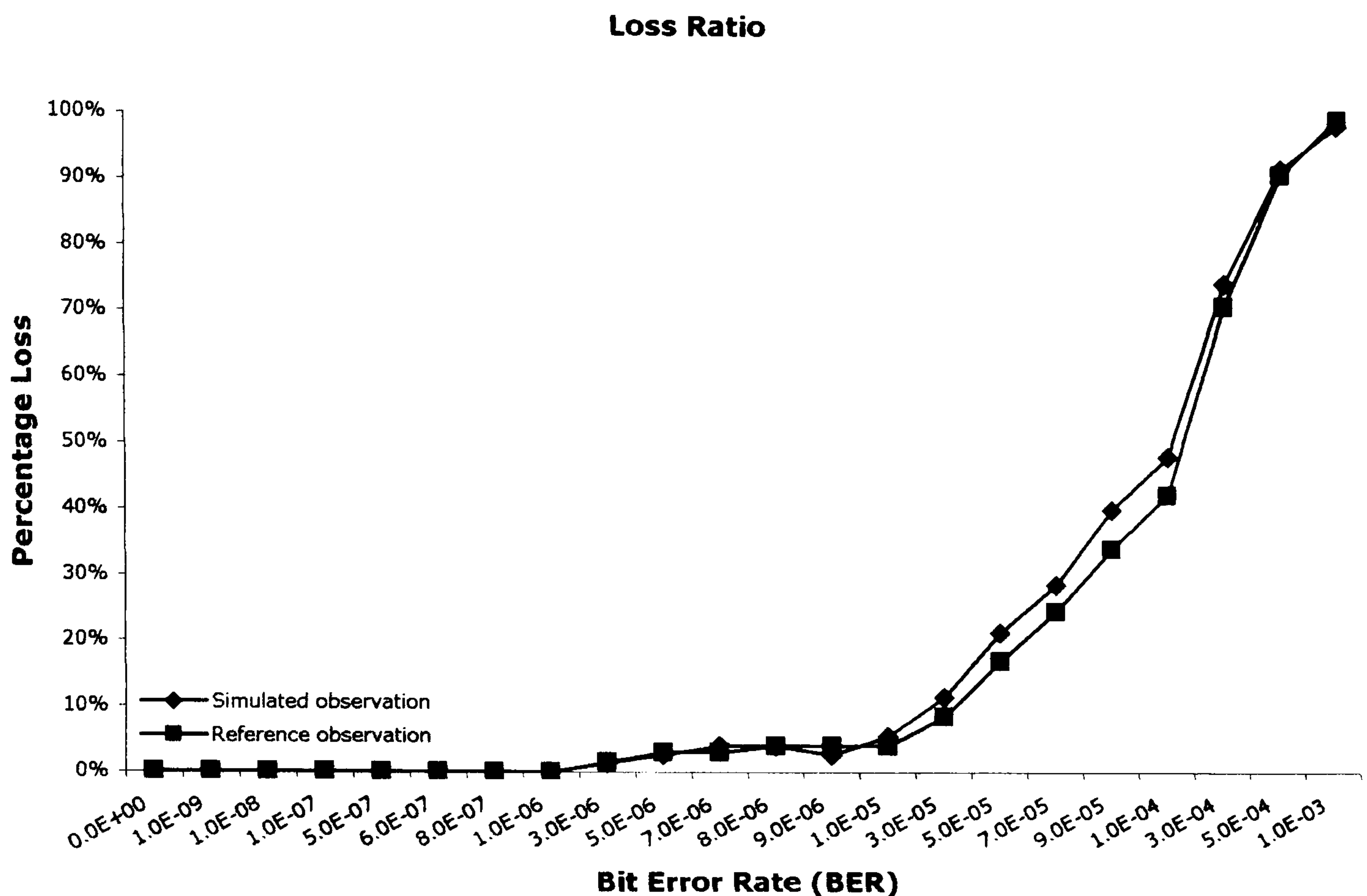


Figure 8.7: Loss ratio – RTT 2.5 msec and variable BER

In case the transmission rate needs to be changed and therefore the protocol timer values, the FCNSEP should be employed, following the identification of the condition and its signalling to the SL. Possible implementations of the FCNSEP should add up to the overall data transfer latency, due to the processing times required for the error protocol messages.

To adequately observe the effects of timer changes in the FCNS throughput response, the following categorisation has taken place, given a frame block consisting of 32 individual frames and a Go Back N implicit flow control mechanism. Congestion has been assumed to be the reason for the FCNSEP realisation running on an interlayer signalling scheme. The effects of the peer error signalling case are discussed in Section 8.3.2.

1. *PHYS layer accepts 32 packets in its buffers before proceeding in creating the FCNS frames. Application instance issues the NODE_ALERT. Assuming a processing delay of 2 msec for every interlayer message, the FCNSEP implementation would result in a 6 msec delay for the FCNS instance to be notified*

for the recommended action (2 msec is also the time that SL would take to reach a decision). If 2 msec is the time that it would take the particular FCNS communication layer to process the request, then the overall time it will take for the user data messages to be sent under the new timer scheme will be 8 msec. Given that a single FCNS frame can be transmitted in 1.24 msec, then the alteration will result in a loss of 6 messages. Due to the particularity of the FCNS frame creation, a whole block will have to be retransmitted towards the PHYS layer leading to a message loss of 32 frames (39.68 msec) and 32 additional ones from the previous frame block that will be lost in transit due to congestion or rejection by the receiver (if transmission is continuous). Therefore, the theoretical overall delay will become 79.36 msec.

2. PHYS layer accepts a single packet, creates the frame and proceeds to the next message. Acknowledgment is requested for every frame block. Application instance issues the NODE_ALERT. For a 2 msec processing delay, 8 msec will be required for the instance to adjust its message timers. Due to the fact that the source is not required to transmit 32 messages before the frame block is created, the message loss incurred will only be the 6 messages that could be transmitted in the 8 msec time. This is because the PHS layer will reject the messages under the previous sending rate and complete the block with the refreshed ones. The receiver will not understand the alteration, since its timer is greater than the delay of 8 msec and it acknowledges the completed frame block due to the Go Back N scheme.

To provide for a clearer view in the effect the FCNSEP implementation would have on the loss and deliverability ratios, an example is further given for the case of RTT of 60 msec (Section 8.2.2.2.4).

8.2.2.2.2 RTT of 2.5 msec – EE_LAYER Packet Throughput

Figure 8.8 depicts the effects of the timers' introduction in the FCNS model for the EE_LAYER packet sending and receiving rates.

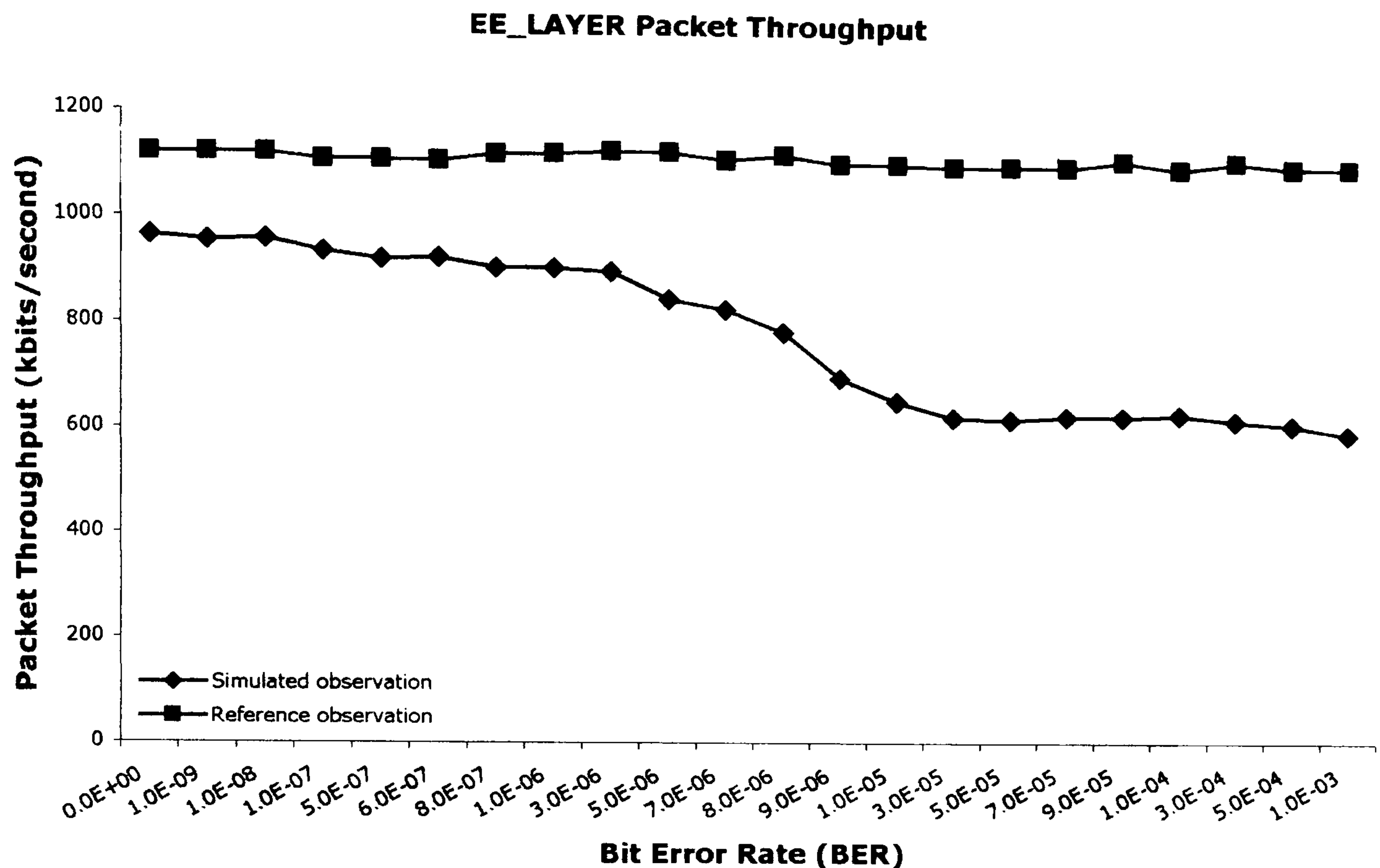


Figure 8.8: EE_LAYER packet throughput – RTT 2.5 msec and variable BER

The reference observation caption represents the FCNS reference packet throughput of Figure 8.5, entailing the ideal values the model could achieve. It is clear from the response of the FCNS that even with the same RTT and BER values, there is a significant decrease as to the amount of packets sent towards the output FCNS instance. This is due to the fact that the measurement is extracted with respect to the simulation time observed throughout the various runs equalling the BER points.

With the introduction of a receiving timer of 50 msec, it becomes apparent that a message loss would expire the timer, resulting in an additional delay of 50 msec plus the time that it would take for the NAK message to reach back to the sender (RTT/2). If the processing time for the NAK is 5 msec for each FCNS instance, then with a 1.25 propagation delay, the latency that would incur will be 61.25 msec before it reaches the sending peer node entity. This implies that if transmission

occurs for the highest T_{1F} , then the loss is translated to *49 frames*, that is, almost one and a half frame blocks.

In Figure 8.8 such variations can be observed for the various BER values. As an example, for a BER probability of 10^{-7} , the loss imposed on the model is of the order of 14 frames, whilst for a representative error-free interval of 2 msec, the packet throughput is further minimised by 39 frames. Given the fact that messages received on unrecoverable errors are discarded, the response of the FCNS system depicted in Figure 8.8 conforms to the protocol specification.

8.2.2.2.3 RTT of 2.5 msec – Frame Throughput

Figure 8.9 depicts the overall frame throughput of the FCNS architecture, which closely matches the one obtained from the reference measurements. Since this observation is solely based on the amount of frames sent and received by the destination, irrespective of the amount of time it took for the messages to reach their recipient, the response of the FCNS follows the design expectations represented by the reference model.

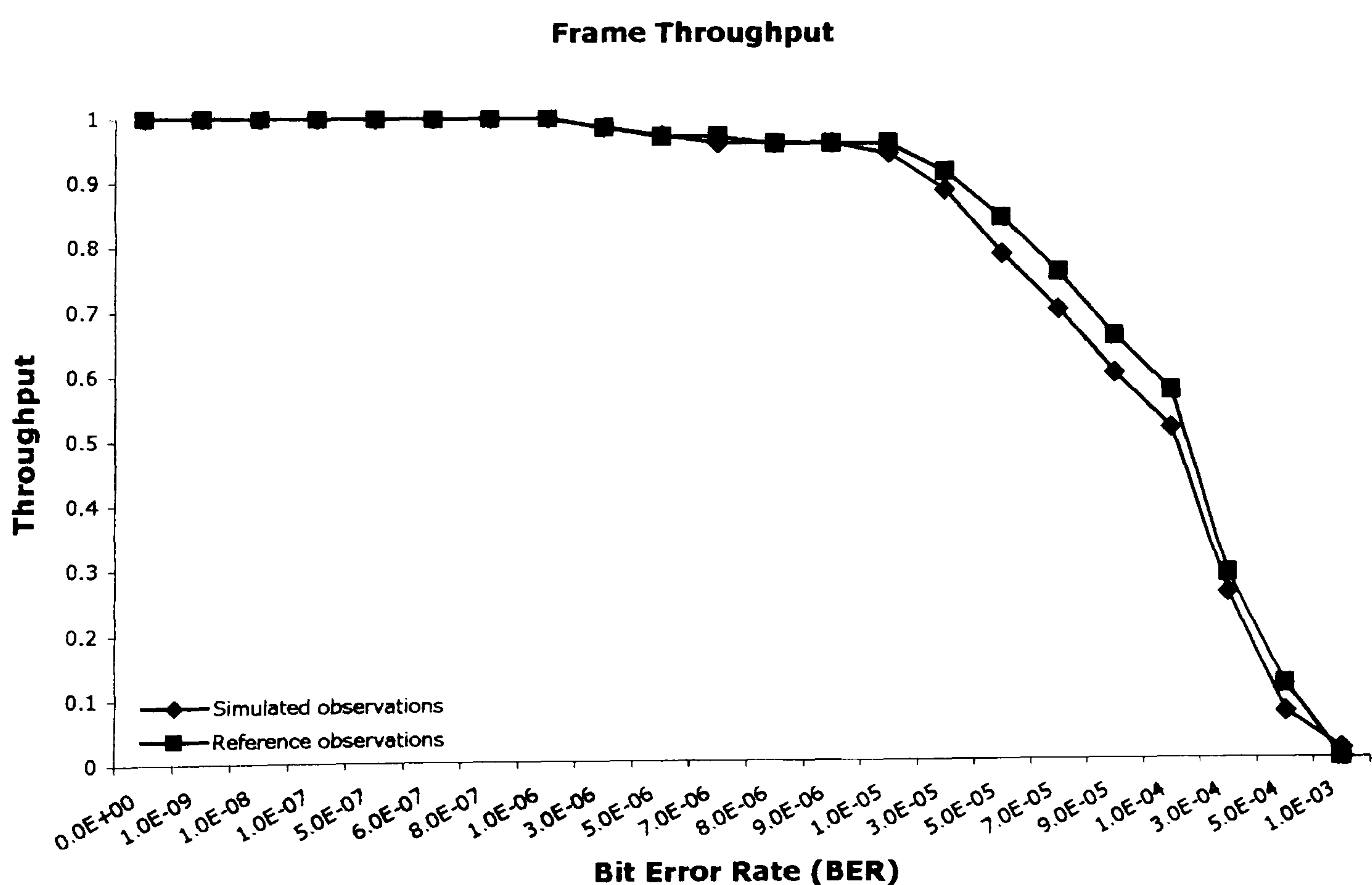


Figure 8.9: Frame throughput – RTT 2.5 msec and variable BER

Following the loss ratio response of Figure 8.7 it can be concluded that the slight variations between the reference and measured FCNS performances are due to the memory consumption deviations of the OMNET++ tool. The application of the protocol timers altered the overall amount of frames in transit, but not the response of the system in detecting and correcting bit errors for the similar delay value of 2.5 msec, resulting in an almost identical deliverability response.

8.2.2.2.4 RTT of 60 msec – Loss Ratio

For these sets of measurements the overall RTT has been increased to 60 msec, keeping as constants the 2 msec message protocol processing delay and the 5 msec FCNS instance processing time for any message sent or received. The introduction of additional latency in the FCNS model has been used to model the response of the protocol in an environment based on a slower communication channel and node instance, increasing the time it would take for the system to apply the FCNS security functions to the user data.

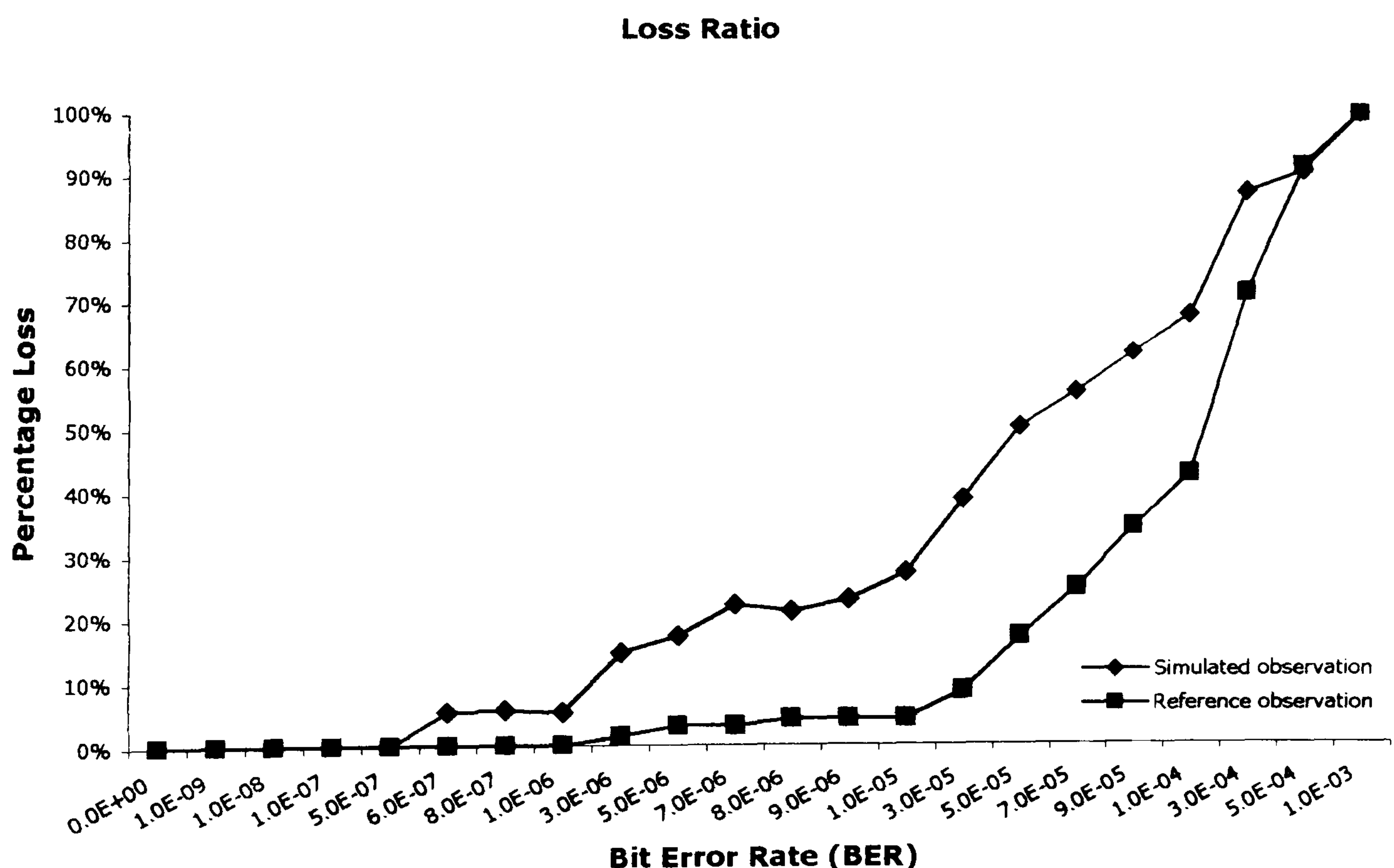


Figure 8.10: Loss ratio – RTT of 60 msec and variable BER

The effects of the additional delay are clear in the loss ratio FCNS response of Figure 8.10. The protocol was unable to maintain a level of frames acceptability

close to the reference observations as the BER was increased. The overall performance of the FCNS stack is acceptable, given the message losses that could occur due to the additional latency and the frames that could ideally be transmitted at that time.

An example erroneous case has been considered for this environment, to enable the identification of the system's response in the application of the FCNSEP. For the interlayer signalling procedure, an initial 25 msec timer had to be changed to adapt to the network environment used for these measurements for the propagation delay of 30 msec. The PHYS layer constructs the FCNS frames as they arrive by the upper layer and does not store them first into a buffer before creating the message block. Figure 8.11 illustrates the effects of the FCNSEP implementation throughout all simulation runs.

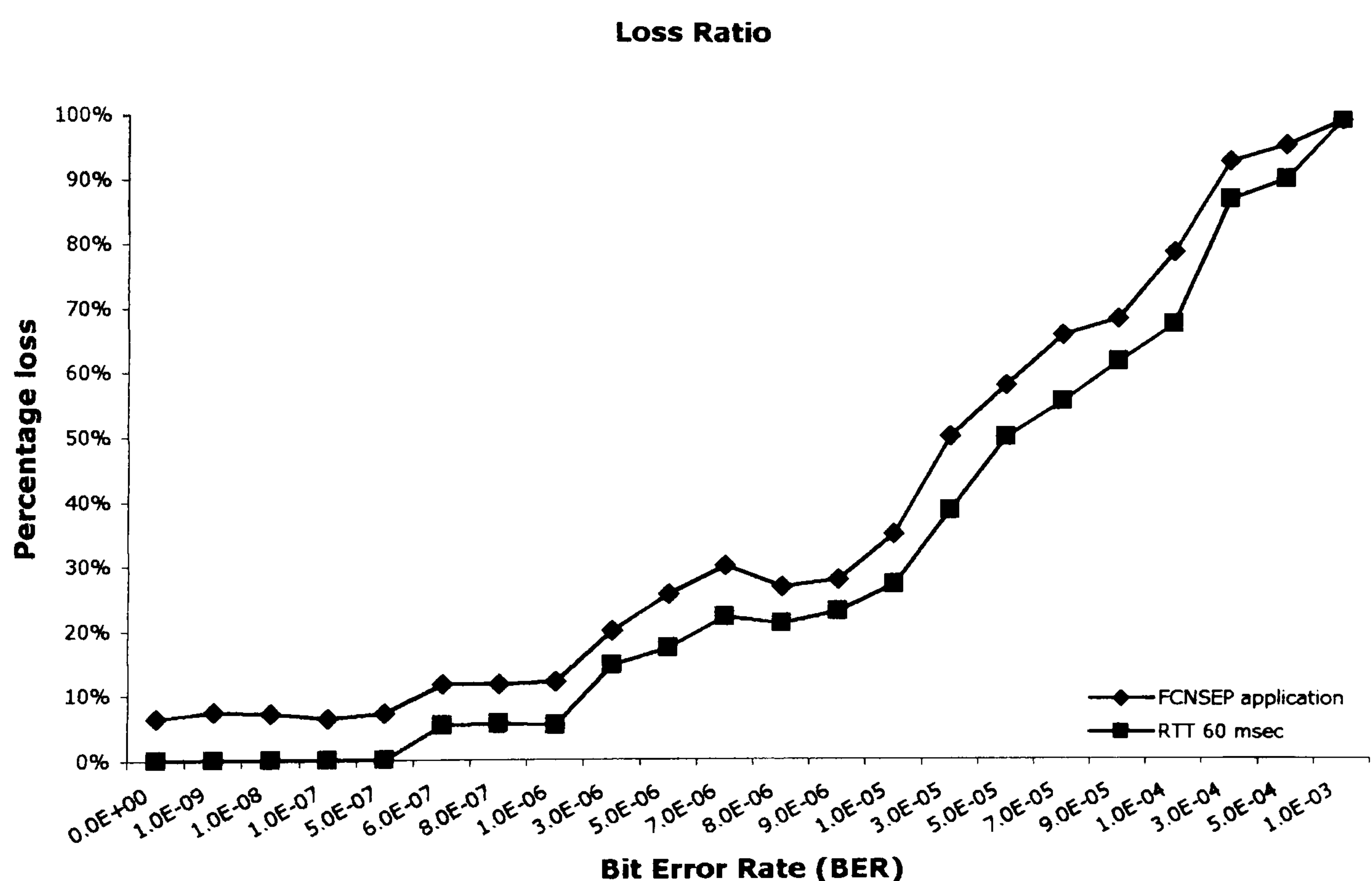


Figure 8.11: Loss Ratio – RTT 60 msec, comparison with FCNSEP realisation

In Figure 8.11 the caption *FCNSEP application* represents the loss incurred in the FCNS system by the introduction of the FCNSEP interlayer signalling messages, whilst *RTT 60 msec* identifies the simulated loss ratio of Figure 8.10.

On a theoretical level, without taking into account the address verification and SC procedures, then the minimum effect of the FCNSEP realisation will be as follows. For the receiving instance to adjust its timer, the NODE_ALERT message is sent after three frames have been discarded, or the receiving timer has expired three times. The time it will take for the SL to issue the appropriate action and the receiver to adjust the new value is 8 msec and hence an overall *13 msec* for the PHYS protocol to be notified of the alteration. For the optimum T_{1F} this should imply a loss of 10 frames, due to rejection by the receiving protocol instance. Overall, the discarded messages should have been *13* from just the alteration process, plus 7 additional frames due to the timer expiration at the PHYS layer before its adjustment ($32\text{messages} \times 1.24\text{msec} = 39.68\text{msec}, 9.68\text{msec} \rightarrow 7\text{frames}$). However, due to the discrete-event nature of the simulator, the loss induced on the system has only been 5 frames (3 that were lost before the FCNSEP application, 1 after the ERROR_RESP reception and 1 after the adjustment of the timer). This is due to the fact that throughout the FCNSEP signalling procedures, the sender has sent no messages as would have been expected in a real-network situation.

8.2.2.2.5 RTT of 60 msec – EE_LAYER Packet Throughput

The effects of the RTT increase with respect to the reference model can be observed in the response of Figure 8.12, which shows the throughput observations for RTT of 60 msec and 2.5 msec compared to the ideal case. The throughput is much reduced by the increased RTT, which results in message losses with respect to the highest MFS.

The introduction of the additional *28.75 msec* propagation time, entails a message loss of 23 messages in relation to the optimum T_{1F} . The losses induced in the simulated model appear greater and of the order of *27 packets* for a representative BER of 5×10^{-7} and *35 packets* for an error-free interval of 100 msec with respect to the RTT of 2.5 msec measurement (timers enabled).

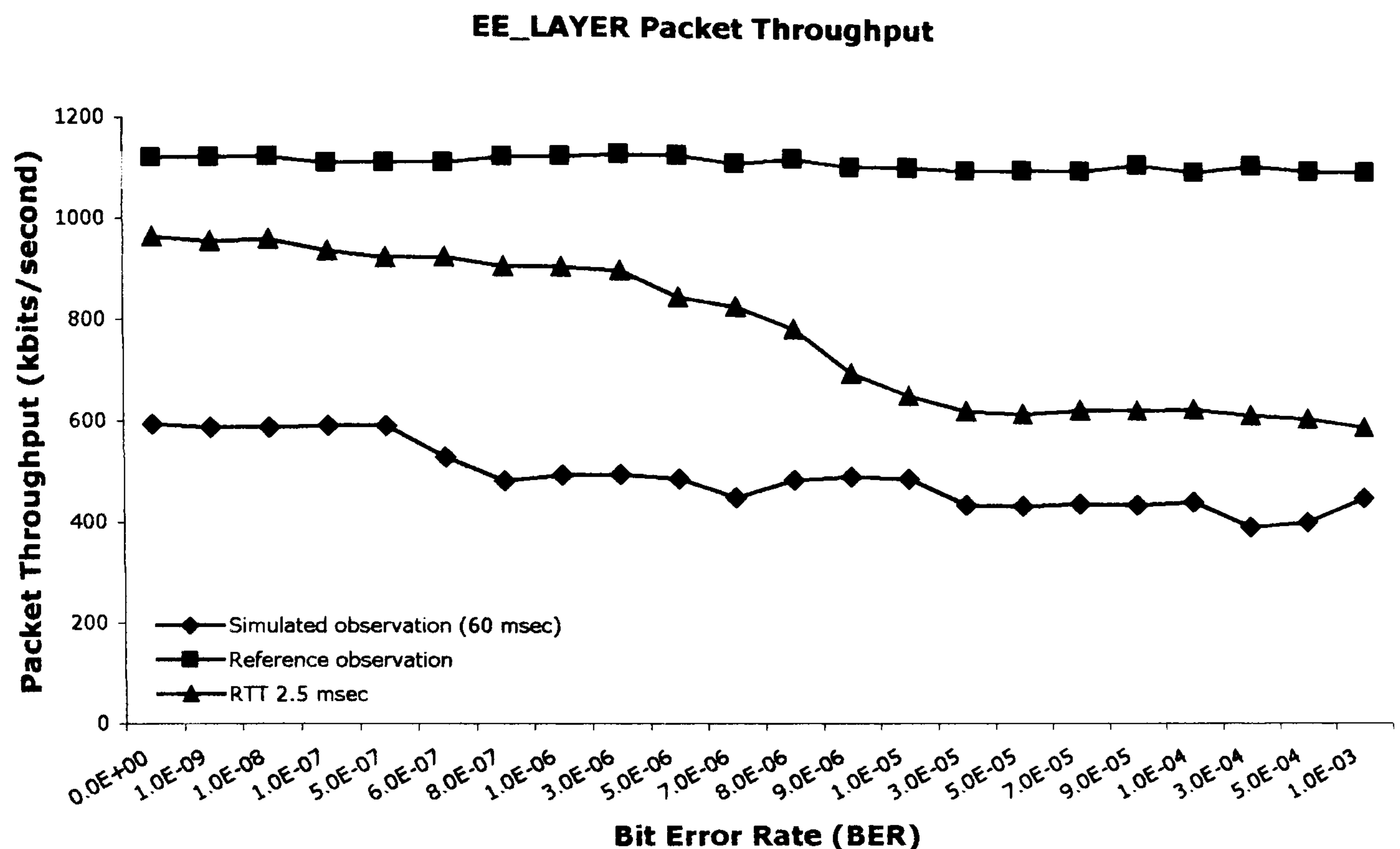


Figure 8.12: EE_LAYER packet throughput response comparisons

The additional loss is due to the FCNSEP application, which as mentioned results in a cost of 5 messages. Given this fact, it can be concluded that Figure 8.12 is acceptable to represent the packet throughput response of the FCNS for the RTT of 60 msec.

8.2.2.2.6 RTT of 60 msec – Frame Throughput

Figure 8.13 depicts the observed frame deliverability ratio in comparison to the reference value.

Due to the added network latency, it is normal for the system to reach lower throughput values than the theoretical measured ones. Overall the system behaved as expected, even in cases where FCNSEP had to be realised to adjust timer settings at the receiving FCNS instance.

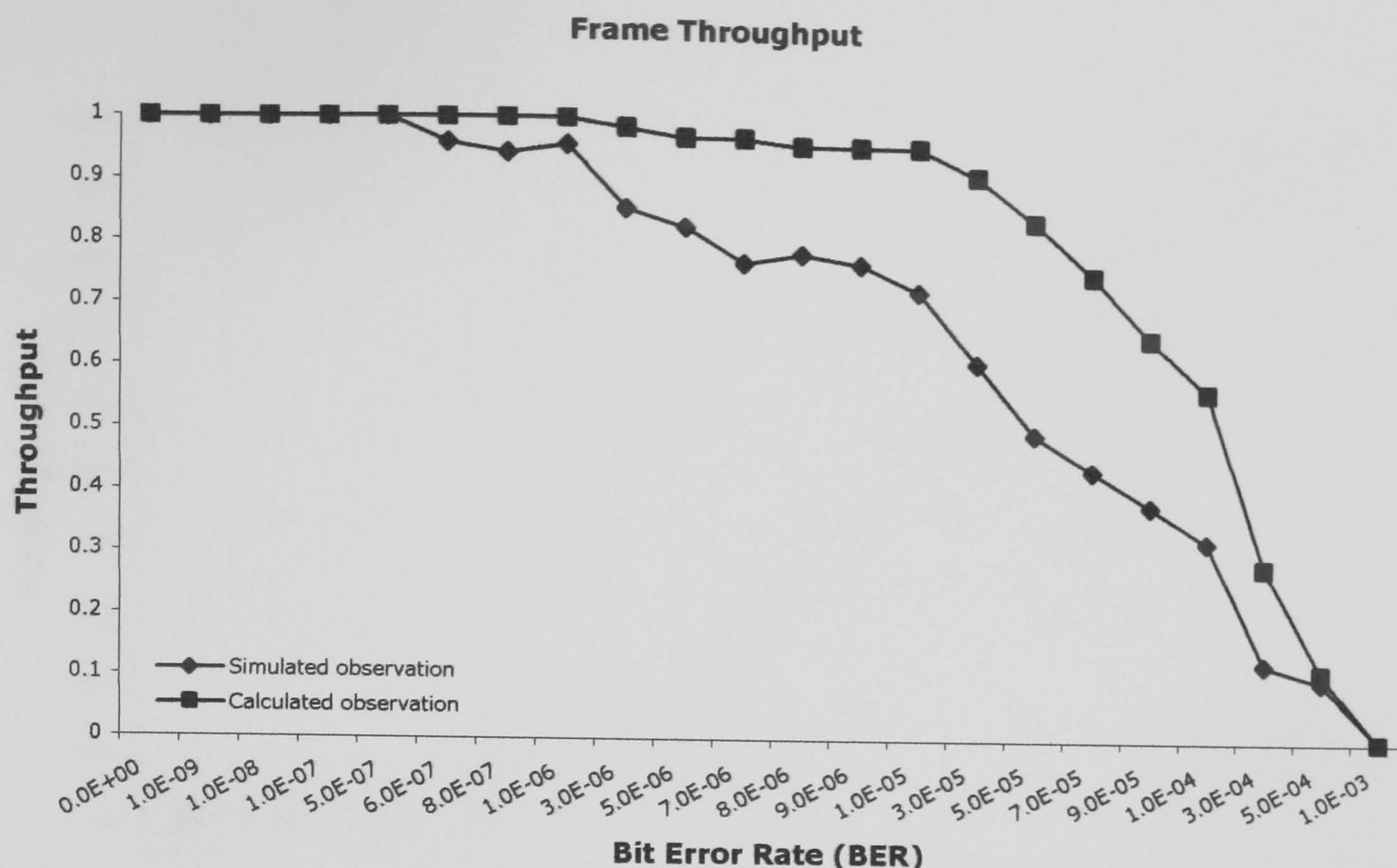


Figure 8.13: Frame throughput – RTT 60 msec and variable BER

8.2.2.3 Variable Delay, Variable BER

For this set of measurements, both the delay and BER have been varied, to enable the identification of the throughput response of the FCNS system. Table 8.4 depicts the jitter imposed on the system with respect to the simulation time.

Table 8.4: Delay variations (jitter) with respect to OMNET++ timing

Simulation time t (seconds)	RTT delay (msec)
$0.8 < t < 3$	200
$3 < t < 5$	100
$5 < t < 6$	400
$6 < t < 7$	260
$t > 7$	200

The excessive RTT of 260 and 400 milliseconds have been used to model a network with very slow communication channels, simulating the effects that congestion might have on the frames in transit.

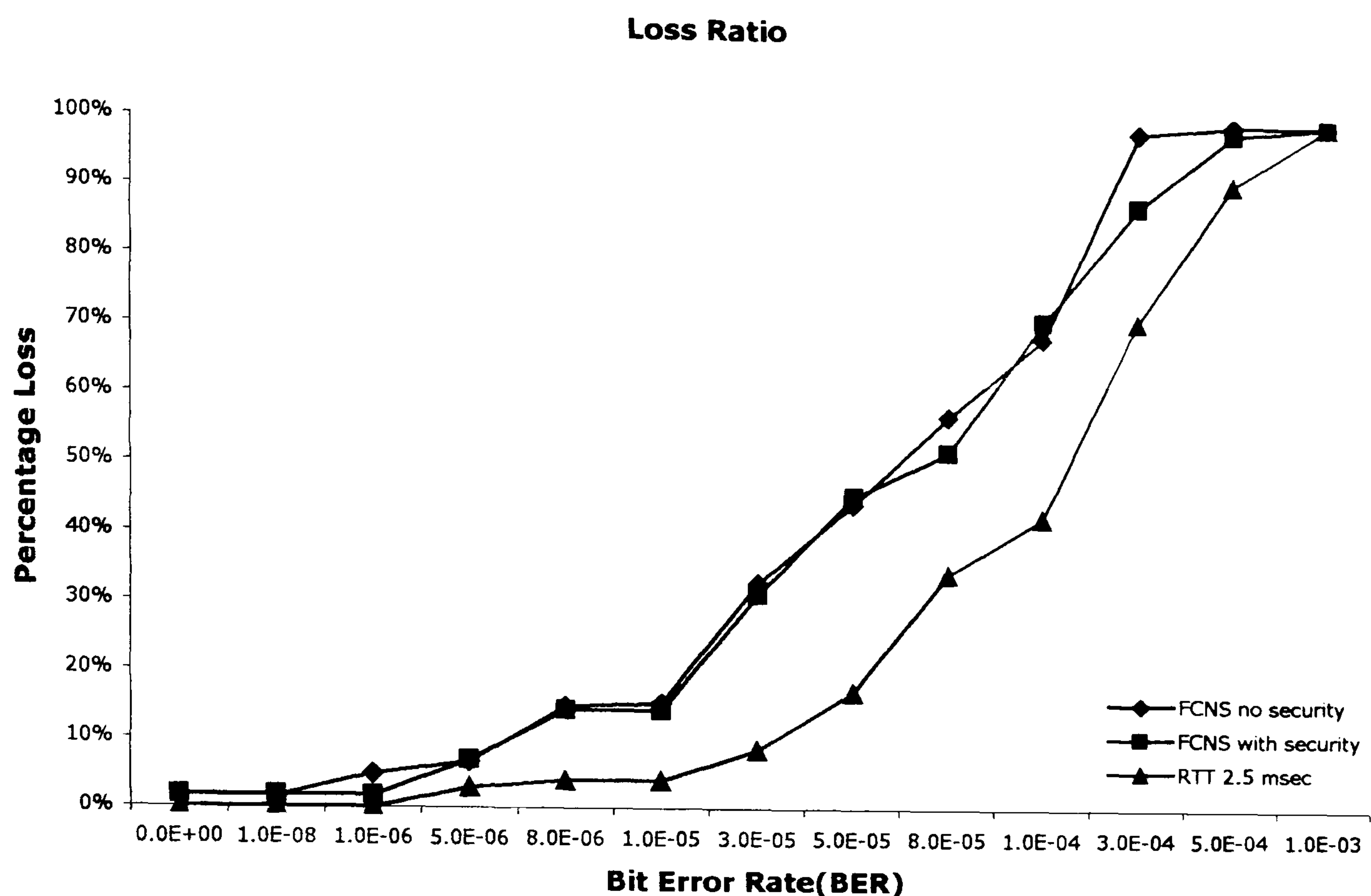


Figure 8.14: Loss ratio – Variable RTT and BER, FCNS realisation comparison

FCNS security caption of Figure 8.14 represents the FCNS system in an environment with full security functionality enabled, whilst *FCNS no security* corresponds to the FCNS system when no security services are required. The latter implies the lack of encryption/decryption techniques and SC exchanges, though the address verification procedures of the UDSES layer have been preserved since they are independent of the security mechanisms of the SL. Comparison is also given in relation to the reference measurement for the RTT of 2.5 msec. The effects of the excessive delays are clearly observed in Figure 8.14, which illustrates the performance FCNS exhibited in heavy faulty transmission conditions.

Additionally, it can be seen that the loss ratio of both settings is very similar, as was expected because of the design considerations of the FCNS stack. The security procedures affect the overall delay of the system in producing and transmitting the FCNS frames and not the way the messages are communicated between the peers. Therefore the throughput of the system is not influenced in the sense of the loss and deliverability ratio, given in Figure 8.15.

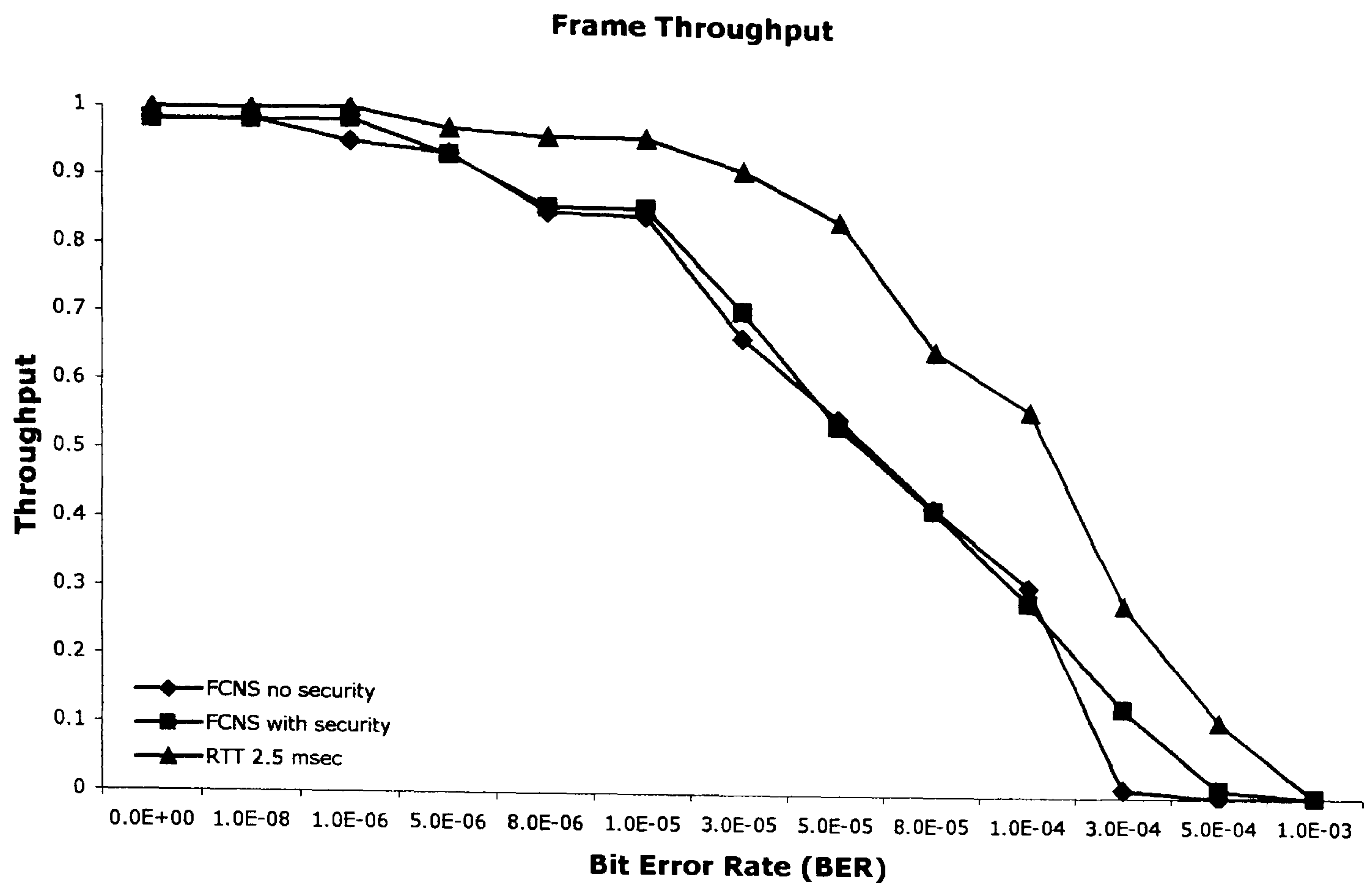


Figure 8.15: Frame throughput – Variable RTT and BER, FCNS realisation comparison

The support offered by the protocol stack is the same, irrespective of the security functionality requested for a particular connection. The differences between the two environments are identified in terms of the data bits in transit and more specifically the relative FCNS efficiency.

8.2.2.4 Relative FCNS Efficiency Measurements

Protocol efficiency measurements have been used to enable the recognition of the differences between a secured FCNS solution and the realisation of the protocol stack with no security mechanisms.

8.2.2.4.1 FCNS Efficiency – Variable Data Size, Constant BER and RTT

Figure 8.16 represents the efficiency of the FCNS architecture for both cases of its implementation. It can be seen that the response of the protocol with full security measures is very close to the one with no security functionality, for a representative error-free interval of 125 msec and an RTT of 20 msec.

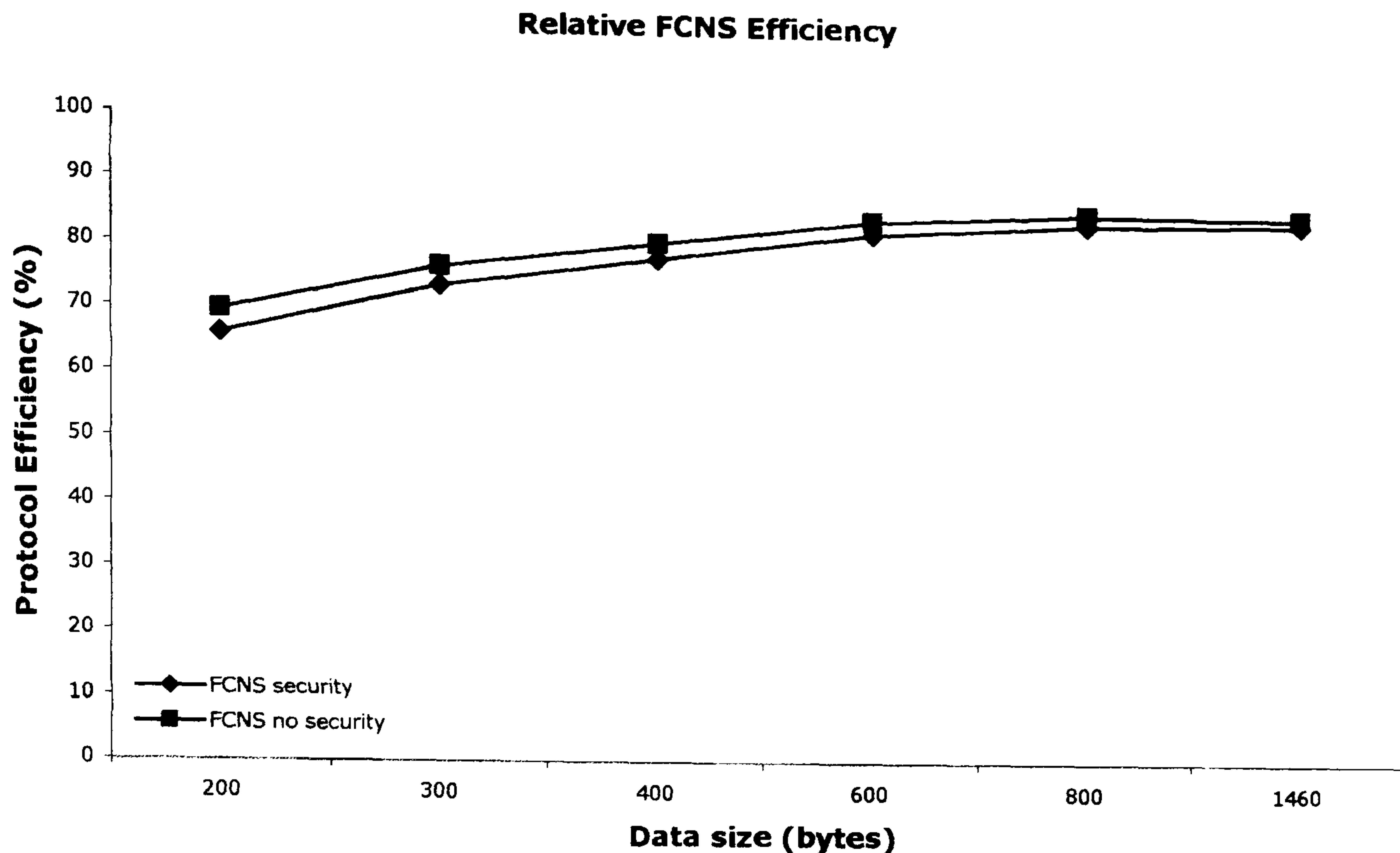


Figure 8.16: Relative FCNS efficiency, FCNS realisation comparison for variable data size

This implies that the system can indeed provide the required security services without loss of data transfer functionality or a significant decrease in its performance. To further enable the measurement of the FCNS efficiency, the simulation environment settings have been altered to provide packet data services for a constant frame size with varying BER.

8.2.2.4.2 FCNS Efficiency – Constant Data Size and RTT, Variable BER

The data size has been kept constant and equal to *1546 bytes* or *12368 bits* for the case of FCNS implementation with full security measures, and *1531 bytes* or *12248 bits* for the FCNS system with no security functionality.

Even for a constant data size and variable bit error probability, the performance of the FCNS for both cases has been almost identical. This notion strengthens the application of full security measures for the FCNS stack, since their employment has a minimal effect on the operation of the protocol.

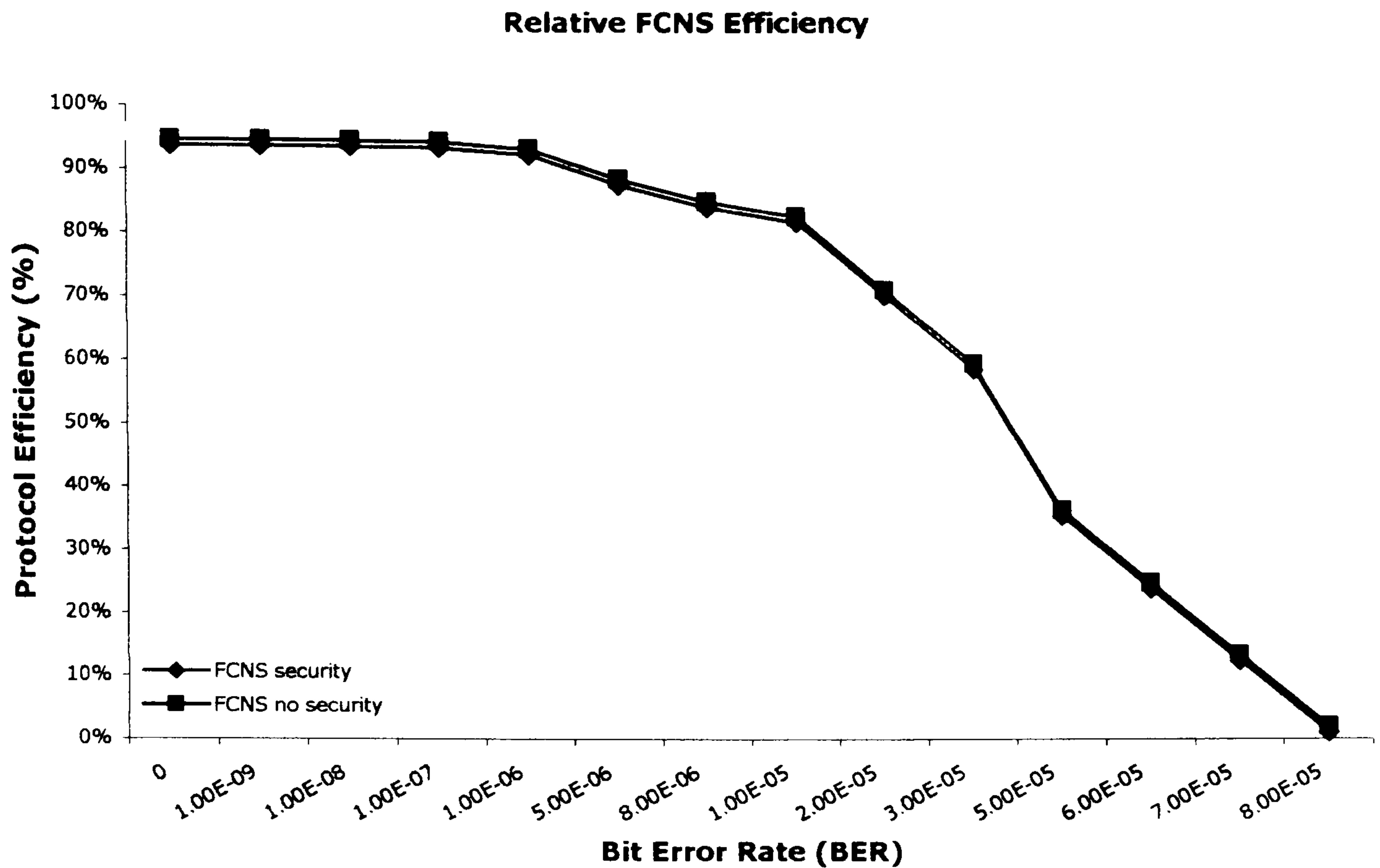


Figure 8.17: Relative FCNS Efficiency, FCNS realisation comparison for variable BER

It can therefore be concluded that the application of the security mechanisms in the FCNS architecture should be prioritised, given the flexibility of the FCNS in adequately adapting to the effects security functions may impose.

8.2.2.5 Constant BER, Variable Delay

These particular sets of measurements explored the behaviour of FCNS when the BER was kept constant, with varying RTT delay. Depending on the value of the error free interval, the following observations have been made to identify the FCNS performance when running on links building up congestion. These results differ from those in section 8.2.2.3 as delay variations did not occur during the simulation run. Additionally, a constant BER has been assumed throughout the delay variations.

8.2.2.5.1 BER greater than 10^{-6} – Loss Ratio

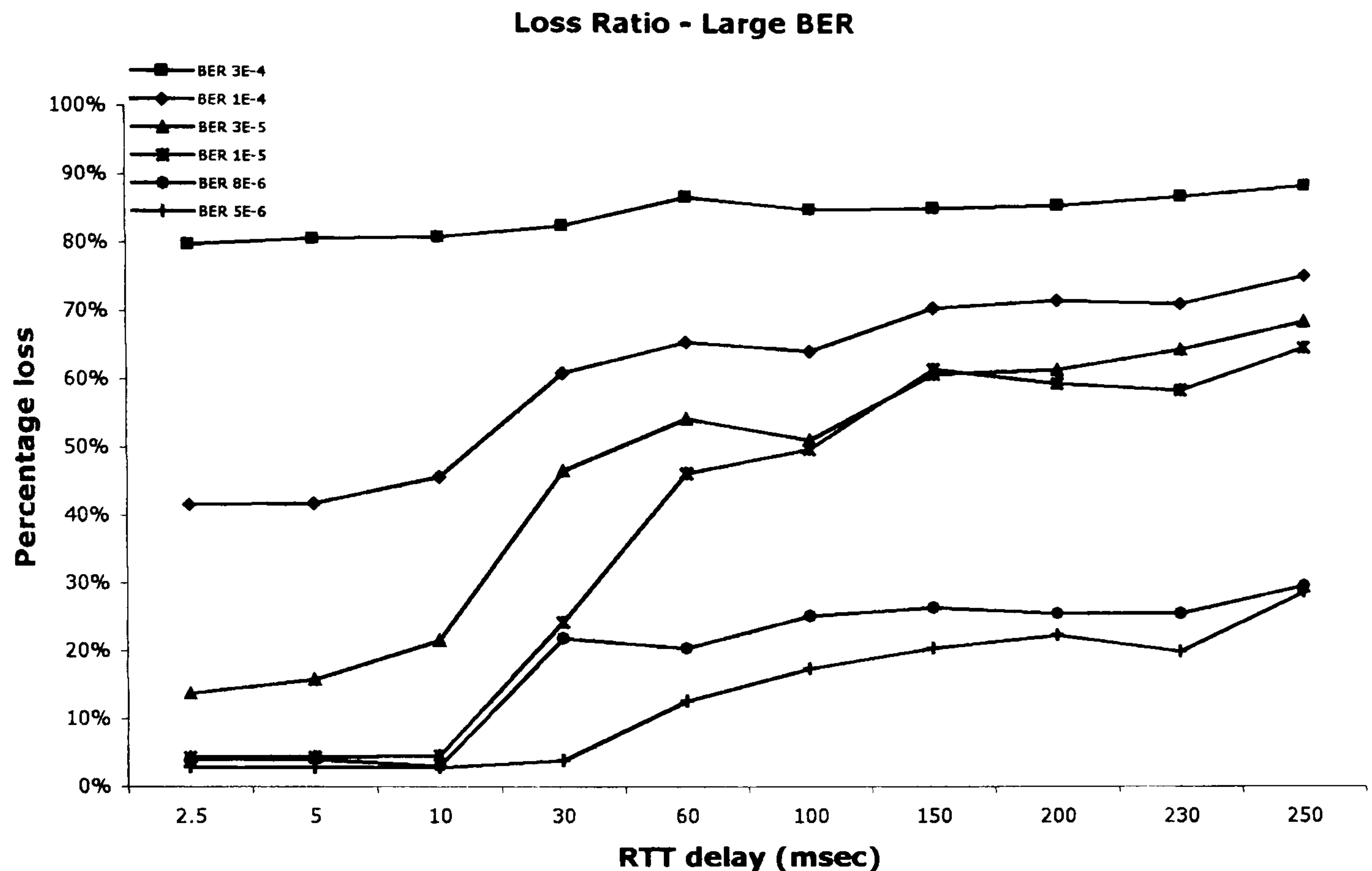


Figure 8.18: Loss Ratio – Variable delay, BER greater than 10^{-6}

Figure 8.18 represents the amount of frames lost during the observations, for communication links becoming congested and with an increased BER imposed on the FCNS frames. For error free intervals greater than 200 msec, the loss ratio of the FCNS maintains a value of less than 5%, that is a loss of 1-3 frames, even with an RTT of the order of 250 msec. As the BER arises, more frames are discarded at the receiver, which also delays the reception process by attempting to correct the bit errors. While the error correction process takes place, further time is lost that could be used to transmit more frames. Additionally, for the PHYS flow control algorithm, message loss implies the retransmission of the messages on error, imposing increased latency on the system that will have to appropriately place the retransmitted frames into the frame block for its recreation at the receiver. This process adds up to the lost frames with respect to the reference measurements for the RTT of 2.5 msec, eventually leading to losses of up to 80% for error free intervals less than 3 msec.

8.2.2.5.2 BER greater than 10^{-6} – EE_LAYER Packet Throughput

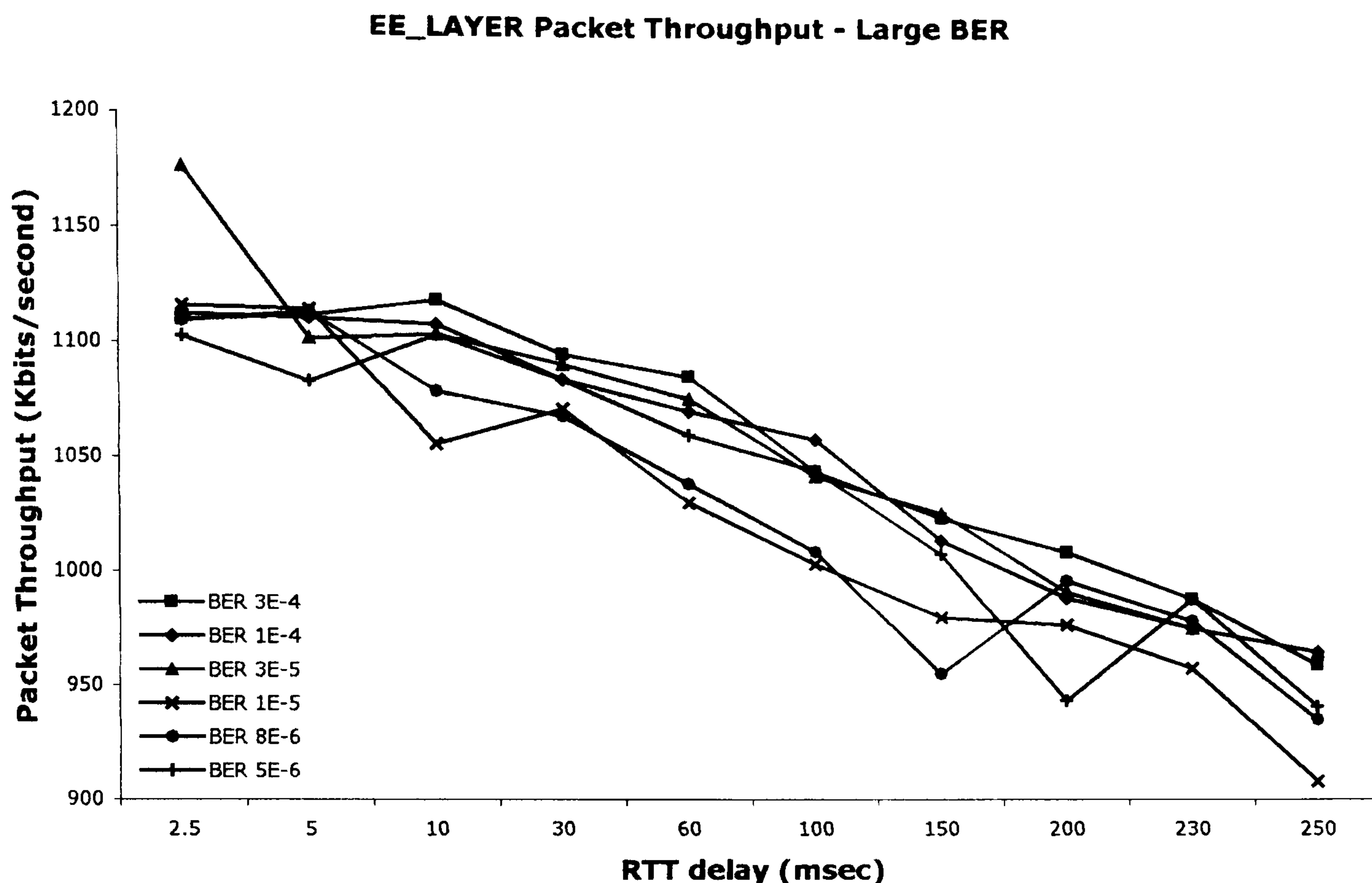


Figure 8.19: EE_LAYER packet throughput – Variable delay, BER greater than 10^{-6}

Figure 8.19 depicts the packet throughput response observed for the duration of the simulation runs. The amount of packets in transit follows the design expectations of the FCNS architecture measured per simulation time second. As the RTT increases, the packet transmission rates fall due to the loss incurred by the added network latency in relation to the highest MFS.

The dependence of the FCNS response to the BER variations is also clearly visible, since small error-free intervals result in more erroneous frames arriving at the receiver. Overall, message losses fall into the order of 29,793 to 94,250 *bits*, that is, approximately 3 to 8 *frames* for delay variations of 30 msec and 100 msec respectively. Given that the propagation time is 15 msec (RTT/2), then for the optimum MFS that would imply a loss of maximum 12 *frames*. For the 100 msec variation the overall loss that could occur is 40 frames for the highest T_{1F} . Therefore, the response of the FCNS outperforms the theoretical expectations, strengthening the flexibility notion of the FCNS in adapting to RTT variations and congestion built up.

8.2.2.5.3 BER less than 10^{-4} – Frame Throughput

Figure 8.20 illustrates the overall frame throughput of the FCNS stack environment for varying RTT values. For bit error probabilities below 10^{-7} , the FCNS achieves virtually 100% deliverability, implying an unaffected from bit errors system. This conforms to the design expectations set in Chapter 7, for which the protocol should provide for an errorless data transmission for small BER values. As the error probability increases, the system has to adapt in network situations, whereby FCNS messages may be discarded due to the modifications occurred in their bit patterns.

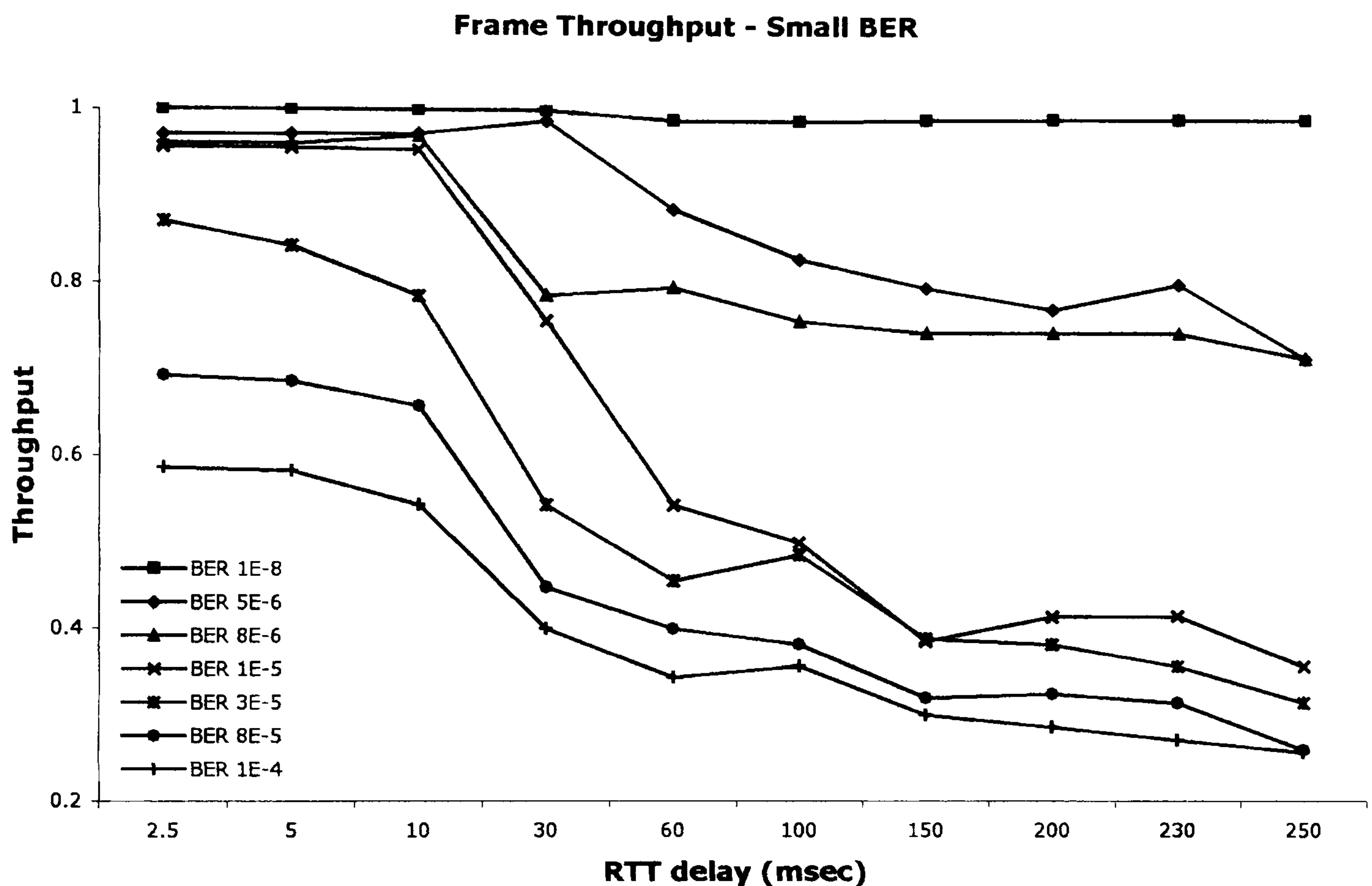


Figure 8.20: Frame throughput – Variable delay, BER less than 10^{-4}

The throughput response provides an adequate measure of the applicability of the FCNS for time and delay sensitive data. It can be concluded that the FCNS can efficiently be used for packet data transmission under erroneous communication conditions. Migration of the FCNS stack into the OPNET environment may provide for a more thorough testing of the protocol procedures, provided that module memory utilisation is far less than that of the OMNET++ tool.

The FCNS protocol stack has been tested against the specification and design considerations presented in Chapters 4 to 7. The protocol simulation environment has been realised to identify the way protocol procedures would adapt to various real network conditions and to test the performance of the protocol in relation to the specified parameters and expectations set in the design process.

The measurements presented in the preceding sections strengthen the applicability of the proposal developed in this work, to serve as the means of an advanced reference architecture for use within packet-switched environments. The implementation of the protocol architecture into a software entity enabled the provision of a test bed for reference secure architectures, which to our knowledge did not previously exist. Therefore, the work has proven the superiority of the FCNS design to standardised models, such as the OSI security architecture presented in Chapter 3.

The following section provides comparisons of the FCNS protocol efficiency and throughput in relation to the Internet suite. The general packet-switched simulation environment enabled the identification of the impact of the security measures application to the FCNS model and its overall performance against currently used protocol structures.

8.3 Packet – Switched Architecture

The packet-switched model network architecture was created to enable the identification of the FCNS response in handling procedures such as message routing and message security in cases where the communicating peers were not directly adjacent. The response of the protocol was tested against delays varying from 100 to 260 msec, in an attempt to simulate the latency such a network may present. Comparisons are given with the TCP/IP Internet suite to provide the means of verifying the efficiency of the FCNS against currently used architectures.

8.3.1 Simulation Environment Description

To realise the execution of the FCNS in a packet-switched network, the simulation model of Figure 8.21 was created via the OMNET++ tool. To preserve consistency with the FCNS stack simulation model of Figure 8.1, the virtual links between the FCNS communication layers and the SL remained intact, imposing the same amount of processing delay on the communication procedure.

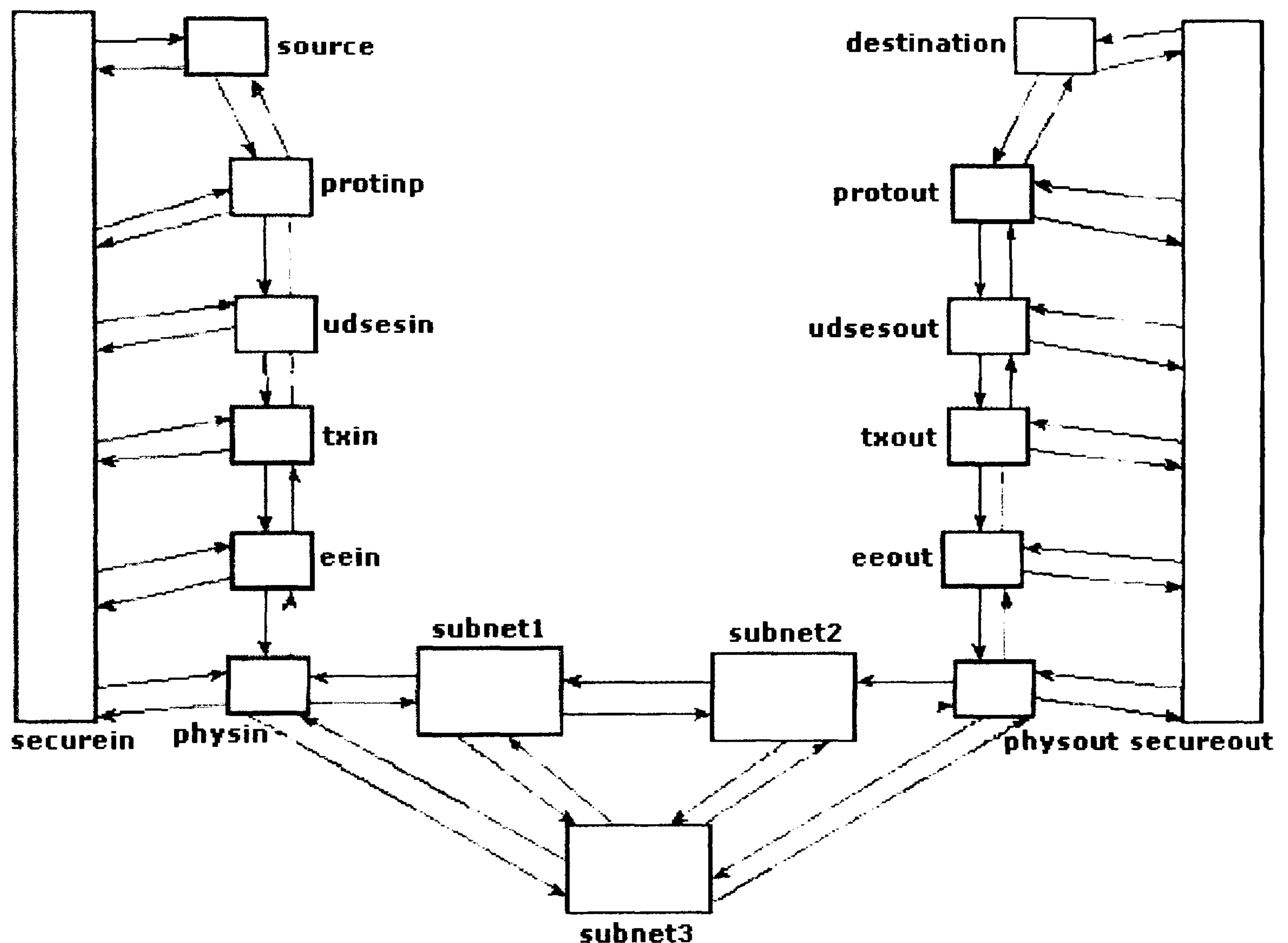


Figure 8.21: Packet-switched network OMNET++ simulation topology

Subnet 1 and *Subnet 2* functional blocks represent intermediate routers running the FCNS subnetwork instances. These include the SL, EE_LAYER and PHYS FCNS layers, to enable the routing and security of the frames exchanged on a realistic network basis. In contrast, *Subnet 3* has been programmed to act as a simple switch, whereby the FCNS frames were simply recognised and forwarded to their destination, without performing any check as to the message protection mechanisms and routing procedures. This subnet has been employed to enable the creation of a secondary communications link, providing alternative routes in case

congestion was ever to arise. For the simulation runs, the primary route has been chosen to be the one following the input FCNS instance through to the subnet1, subnet 2 and the output instance of the stack.

The identification of the particular module blocks was made to enable the abstract representation of the UMTS CN packet-switched domain and address the performance issues of the FCNS in handling services such as the Packet Data Protocol (PDP) exchange procedures and the transmission of signalling data between the CN elements. Chapter 9 identifies the elements of the UMTS CN PS domain and the applicability of the FCNS in delivering the required services.

The parameters used to depict the performance of the FCNS are the ones identified in Section 8.2.2, including the loss and deliverability ratios and packet and frame FCNS throughput measurements.

8.3.2 Results and Measurements

The reference set of simulation measurements was realised for a propagation delay of 50 msec, distributed as 10 msec between the input FCNS and subnet 1 blocks, 20 msec between subnet 1 and subnet 2, and 10 msec between subnet 2 and output FCNS instance resulting in an RTT of 100 msec.

8.3.2.1 Constant Delay, Variable BER, Reference PS measurements

The effects of the varying BER observed for the PS network topology are depicted in the proceeding sections.

8.3.2.1.1 RTT 100 msec – Loss Ratio (reference measurement)

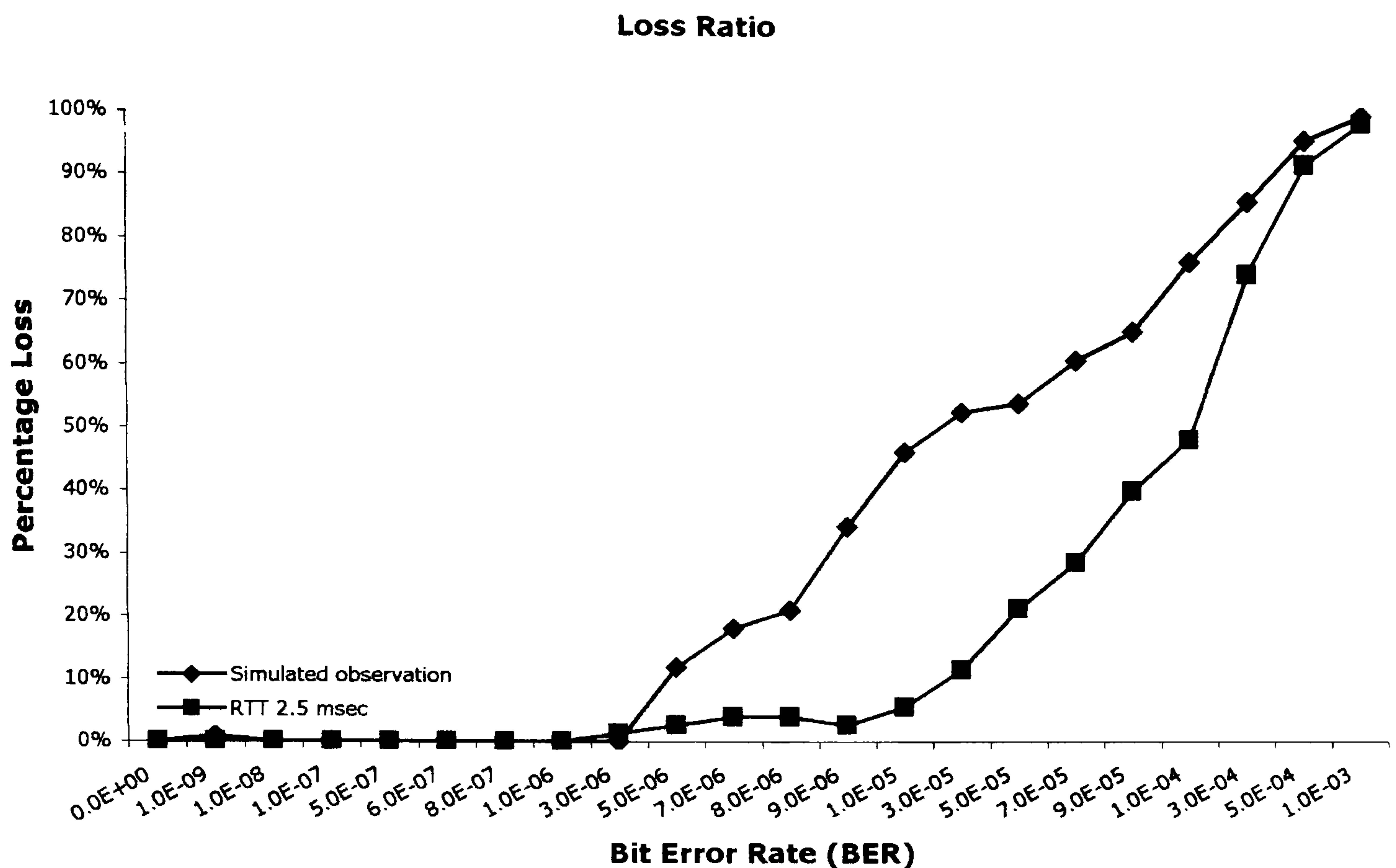


Figure 8.22: Reference loss ratio – RTT of 100 msec and variable BER

In Figure 8.22, comparison is provided with the reference measurement for the RTT of 2.5 msec of the FCNS stack environment. The increased loss ratio is due to the added latency in the network and the presence of the intermediate nodes, which have delayed some of the FCNS frames in their routing towards the destination. This has also been due to the discrete-event nature of the OMNET++ tool, where the destination timers have been expired due to messages not being continuously forwarded to their peer. If the network is further flooded by messages resulting from a sending rate close to the optimum T_{1F} , then the problem becomes more severe, since the simulator is unable to handle the amount of data in transit and at the same time acknowledge the frame blocks on time. However, the response of the system matches the design expectations set for such a topology. Its use has therefore been acceptable as a reference for the PS network environment.

8.3.2.1.2 RTT 100 msec – EE_LAYER Packet Throughput (reference measurement)

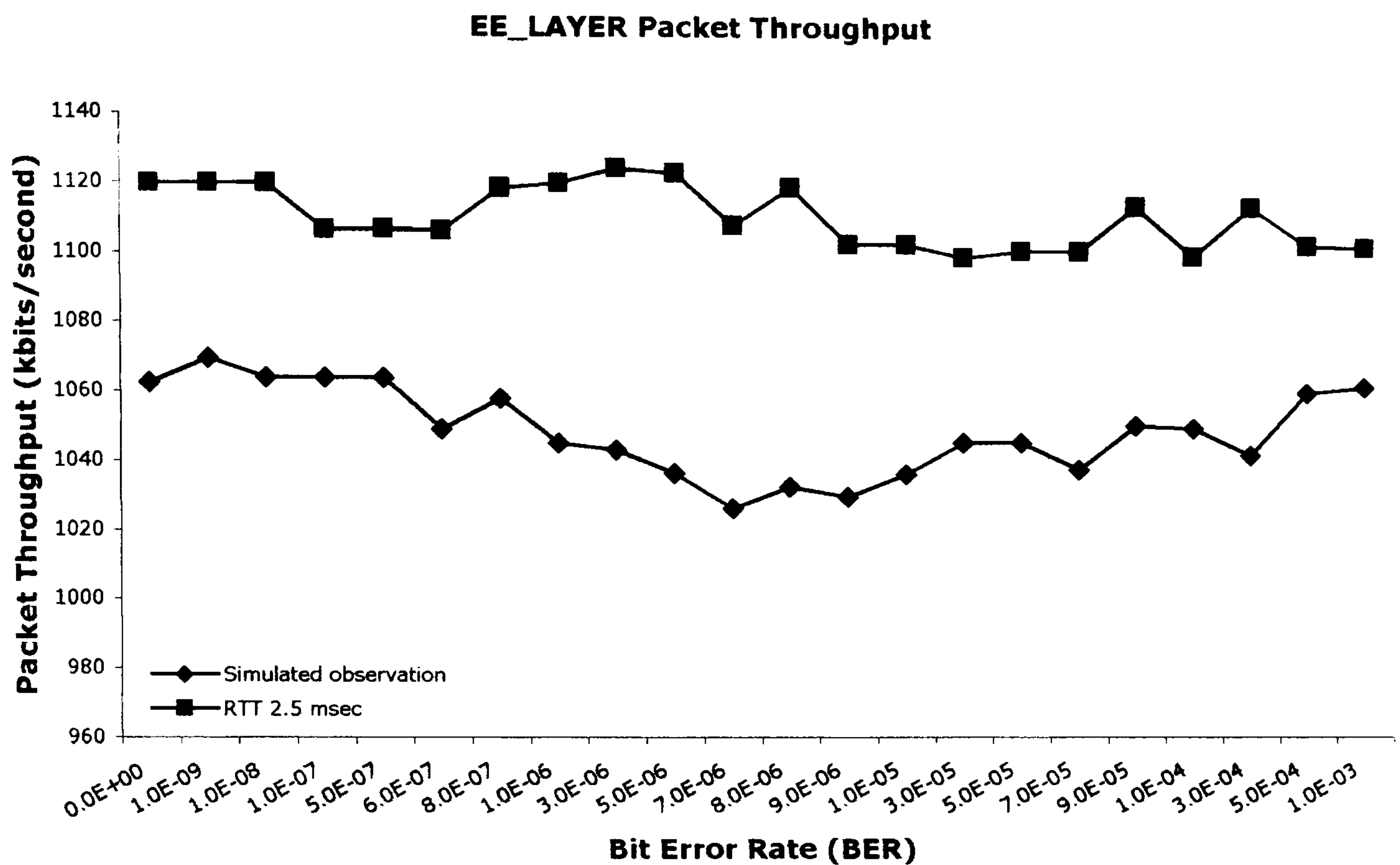


Figure 8.23: Reference EE_LAYER packet throughput – RTT of 100 msec and variable BER

The EE_LAYER packet throughput of Figure 8.23 represents the response of the system in delivering the FCNS frame payload for the given simulation time. The illustrated measurement corresponds to the performance expected by the FCNS, given the RTT of 100 msec. The throughput response relates to a model for which transfer syntax and QoS parameters negotiation completed in two attempts to provide the means of simulating a real network situation. The variations observed in the packet throughput response are of the order of approximately 3 messages, resulted for the additional ones required for the connection establishment parameters negotiation.

8.3.2.1.3 RTT 100 msec – Frame Throughput (reference measurement)

To enable the comparison of the actual deliverability of the FCNS system with respect to the RTT 2.5 msec, Figure 8.24 is used to illustrate the frame throughput for the RTT of 100 msec.

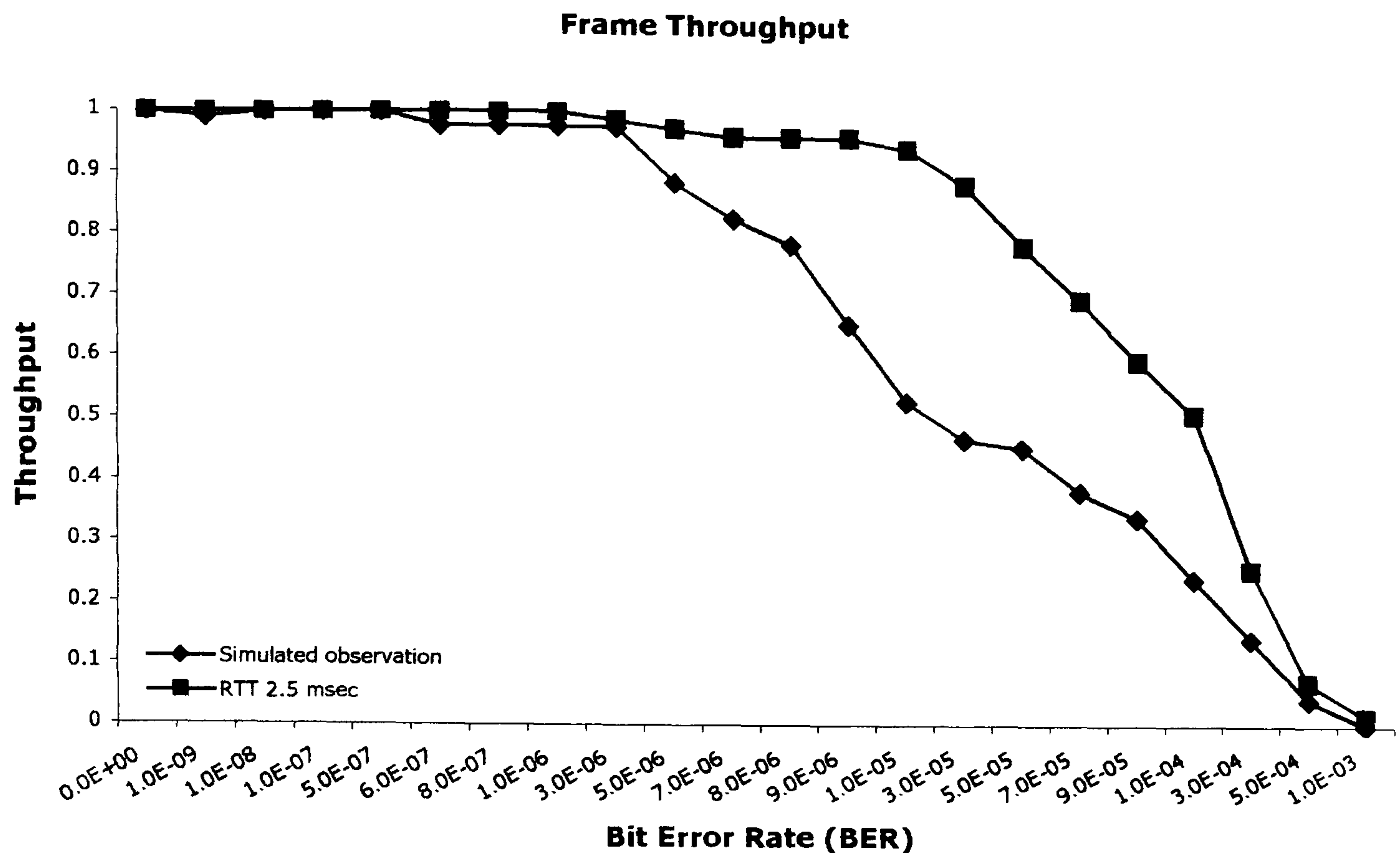


Figure 8.24: Reference frame throughput – RTT of 100 msec and variable BER

Following the observation of the loss ratio of Figure 8.22, it can be seen that the presence of additional network latency has decreased the throughput of the FCNS frames for the PS simulation environment. The response illustrated follows the design expectations, given the increased delay present in the model in relation to the reference measurement for the RTT of 2.5 msec.

For the PS environment, the reference measurements were based on the assumption that the transmission delay was a combination of the message processing delay presented by the FCNS layers and the propagation delay imposed by the communications link. To model a more realistic network topology, the observations given in the following section encompass two additional latency parameters, being the queuing and the insertion delay presented by the intermediate network nodes. The introduction of increased RTT values was also made to enable the simulation of the effects of the SL functionality to the communication process. Functions such as authentication, encryption and decryption introduce additional processing delay for the FCNS protocol instances. Consequently, the parameters used in the proceeding section represent the overall

delay it would be imposed on a message transmitted towards the output FCNS instance.

8.3.2.2 Constant Delay, Variable BER, PS measurements

The results presented in the following sections concern measurements taken for RTT of 200msec and 260 msec. In the former case, 30 msec propagation delay has been imposed to the input FCNS instance - subnet1 and subnet 2 - output FCNS instance links, whilst 40 msec have been allocated to the subnet1 – subnet 2 connection. In the latter case, 40 msec transmission delay has been imposed to the input FCNS instance - subnet1 and subnet 2 – output FCNS instance links, with the remaining 50 msec induced in the subnet1 – subnet 2 one. All links to and from subnet 3 have been assigned a propagation delay of 20 msec for the RTT 200 msec case and 50 msec for the RTT 260 msec case.

8.3.2.2.1 RTT 200 msec – Loss Ratio

Figure 8.25 depicts the behaviour of the FCNS system in delivering the user data in terms of the loss ratio, compared to the reference measurement of the RTT 100 msec. The performance of the system follows the design expectations, bearing small deviations compared to the reference response. The presence of added network latency has increased the amount of lost frames transmitted for bit error probabilities of 10^{-5} and above, due to the error correction mechanisms applied on the received frames and message discarding for BER greater than 10^{-4} .

In such cases, excessive delays and BER may render the data transmission process defective due to the effects communication conditions might have on the information in transit. Furthermore, BER greater than 10^{-5} imply not only the heavy distortion on the FCNS frames, but the probability that more frames may arrive on error compared to error-free intervals of greater than 10 msec.

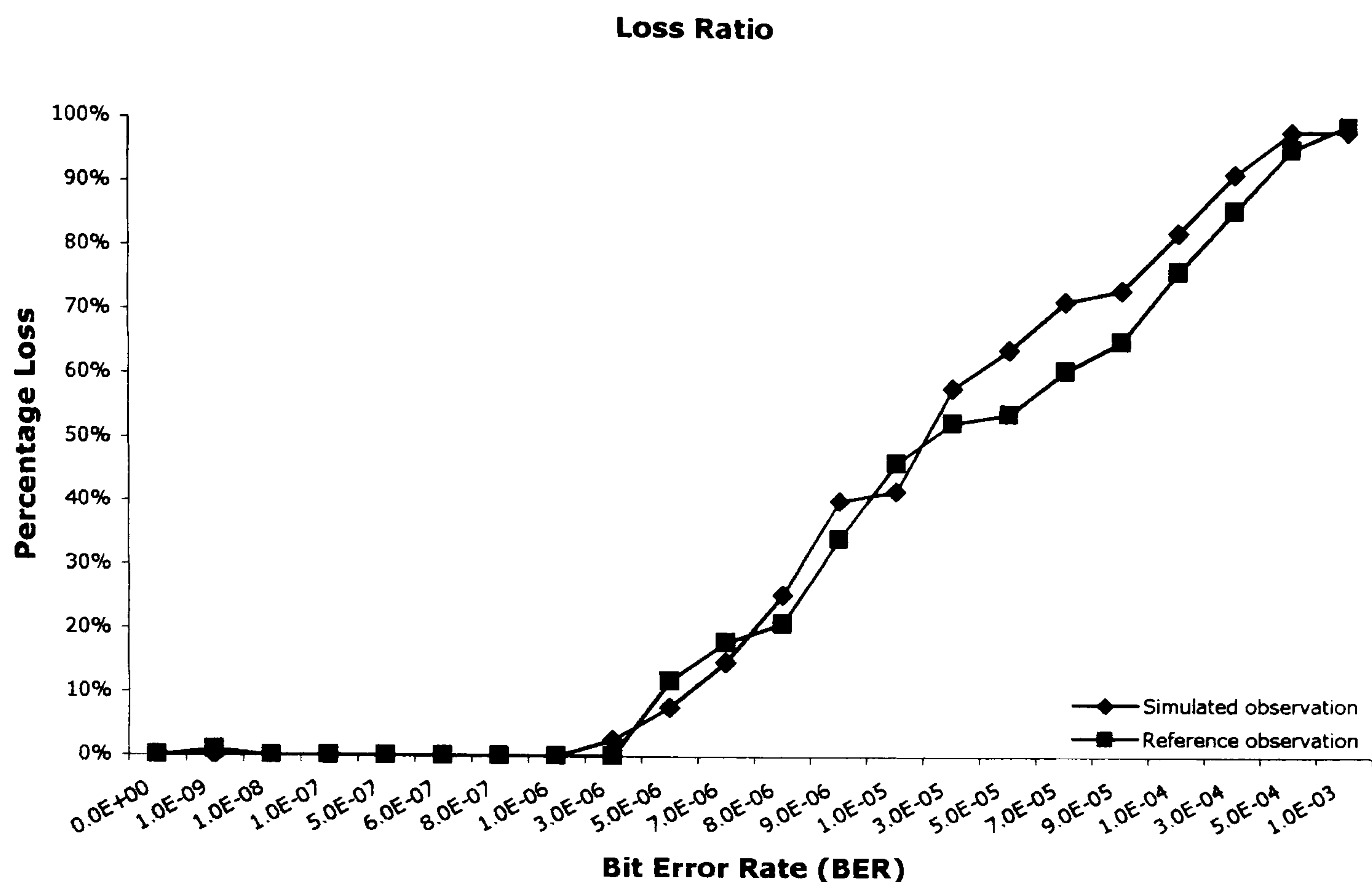


Figure 8.25: Loss Ratio – RTT of 200 msec and variable BER

8.3.2.2.2 RTT 200 msec – EE_LAYER Packet Throughput

Figure 8.26 represents the packet throughput response of the FCNS compared to the RTT 100 msec observation. As is expected, the packet throughput response for the RTT 200 msec is less than the reference measurement due to the doubled propagation delay in transmitting the user data.

For a representative BER of 10^{-6} , the amount of bits in transit was 1,011,537 resulting in a CU of 10.11% for the datarate of 10 Mbps. Given the topology created for the PS environment, it can be concluded that the transmission rates imposed on the system resulted in a congestion-free communication process.

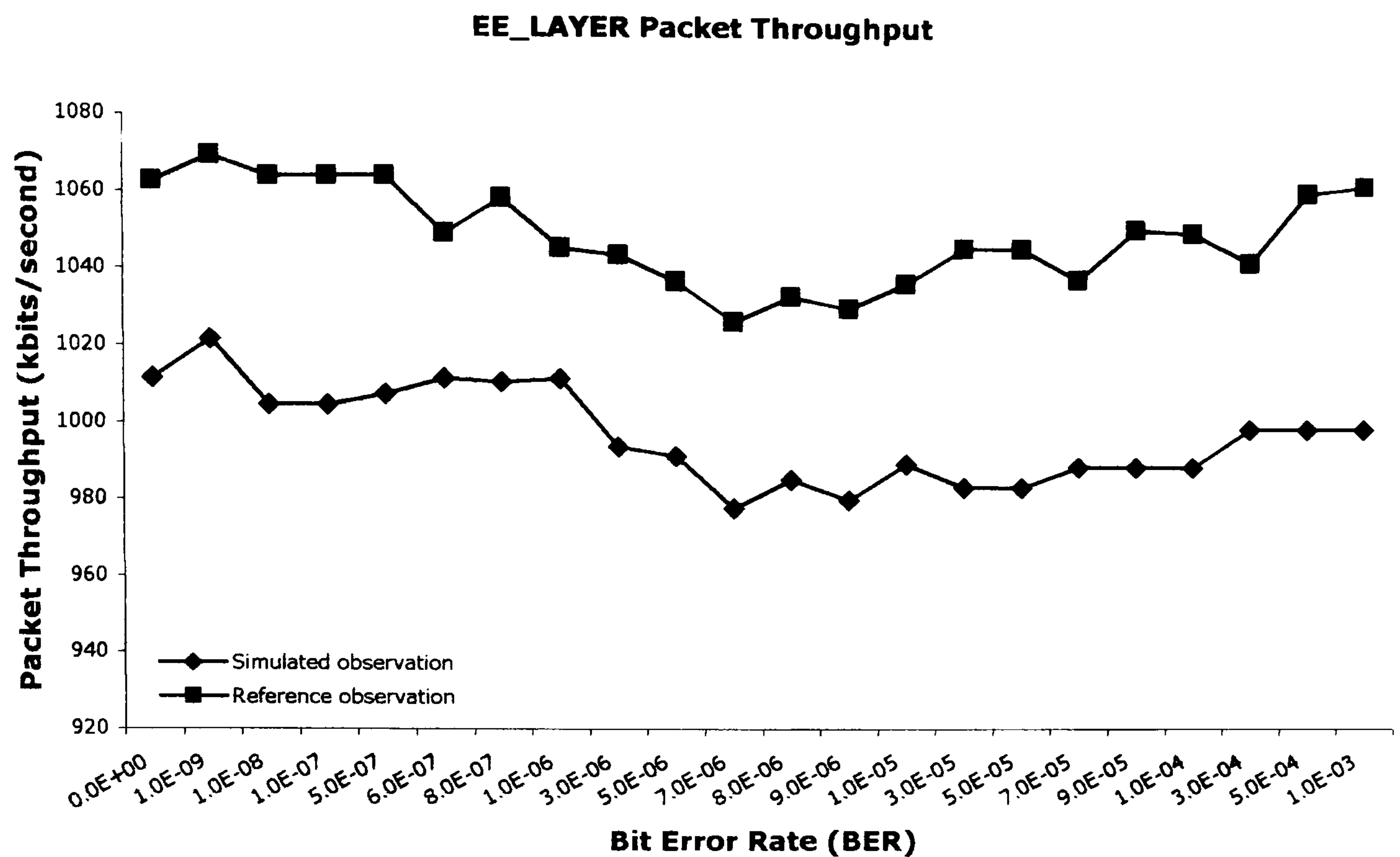


Figure 8.26: EE_LAYER packet throughput – RTT of 200 msec and variable BER

8.3.2.2.3 RTT 200 msec – Frame Throughput

The graph of Figure 8.27 represents the performance of FCNS in detecting and correcting frames arrived on error, for the particular timer instances set for these measurements. The deliverability ratio of the FCNS protocol system for the RTT of 200 msec conforms to the theoretical expectations, given its independence from the simulation time run periods. The behaviour of the FCNS stack follows the principles identified in the preceding sections, provided that the theoretical frame loss for the highest MFS could have been in the range of *40 frames* and above (compared to the RTT 100 msec reference measurement). The applicability of such results is described in Chapter 9, where the settings presented in this section have been used to address service requirements for the UMTS CN packet-switched domain.

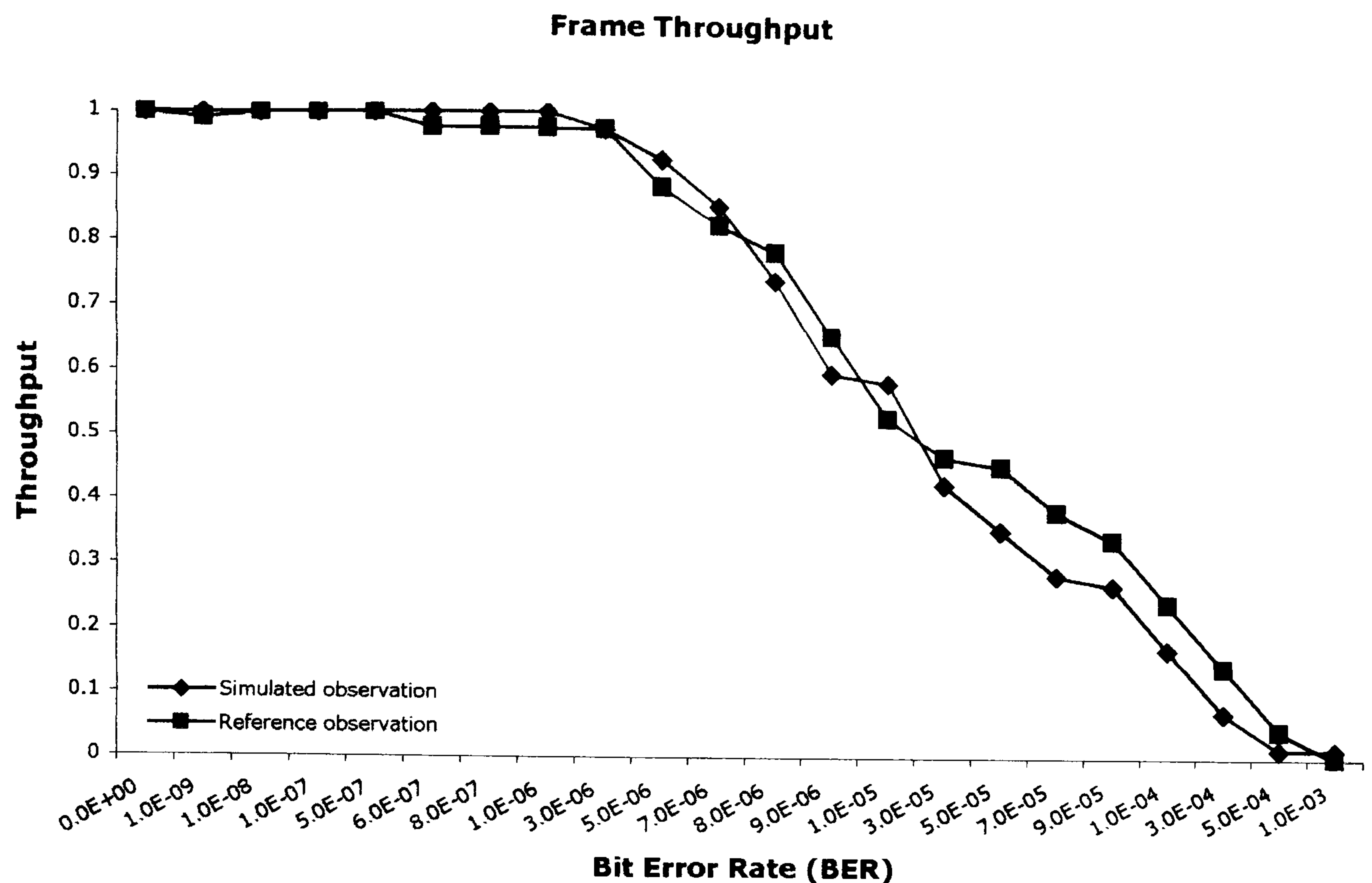


Figure 8.27: Frame throughput – RTT of 200 msec and variable BER

8.3.2.2.4 RTT 260 msec – Loss Ratio

To enable the identification of the FCNS response for higher propagation delays, the work has constructed a set of measurements bearing an RTT of 260 msec. The individual processing times induced to the simulation environment links have been programmed to simulate a long-distance network topology, whereby latency is increased with respect to the distance between the adjacent nodes.

The destination instance message timers have been set to 260 msec (by the FCNSEP peer-error signalling procedures), to observe the response of the protocol in adapting to network environments bearing high transmission delays and BER.

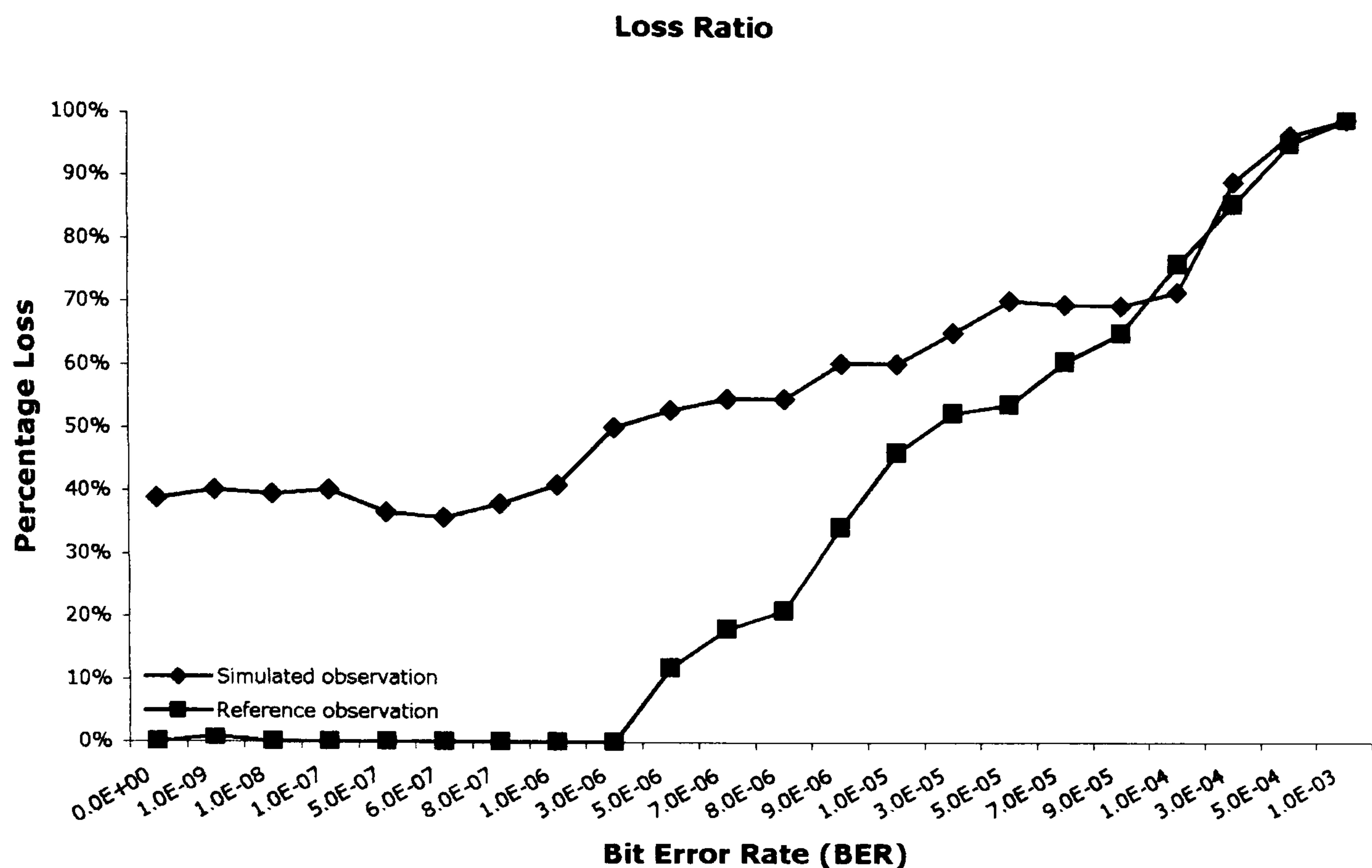


Figure 8.28: Loss ratio – RTT of 260 msec and variable BER

Figure 8.28 depicts the effects of the increased network latency in relation to the destination timers matching the RTT time. Even for very small bit error probabilities the protocol system exhibited losses of 40% with respect to the reference observation of RTT 100 msec. Given that the difference between the propagation times is 110 msec, the losses induced in the packet-switched system match the theoretical expectations and even present greater performance than the ones calculated for the optimum T_{1F} (89 messages).

The percentage loss observed in these measurements can also be explained due to the application of various FCNS functions for the establishment of the data transfer process. As the BER increases, the probability that any message is received on error is maximised, irrespective of the phase communication may reside. Additionally, if the primary route fails, then the transmission of the FCNS messages via an alternative path will result in further losses, due to the path and route verification procedures that must take place between the EE_LAYER and the SL at all network nodes. Furthermore, SC exchanges may need more than two attempts to complete, since BER may affect the contents of the respective primitives at any

node of the simulation environment. Moreover, due to the discrete-event nature of OMNET++, acknowledging service primitives exchange may also delay the transmission of the FCNS frames, due to the amount of simulation time lost. Given that the destination timers should expire virtually after the frame block acknowledgement reaches the input FCNS instance, the response presented in Figure 8.28 represents how efficient FCNS is in providing data transmission services in heavily congested network environments.

It should be noted, that for this particular environment, FCNSEP peer error signalling has been enabled, to increase the destination timers due to the added network latency that may lead to messages losses. Its application has decreased the number of frames sent, due to the discrete-event nature of the simulator.

8.3.2.2.5 RTT 260 msec – EE_LAYER Packet Throughput

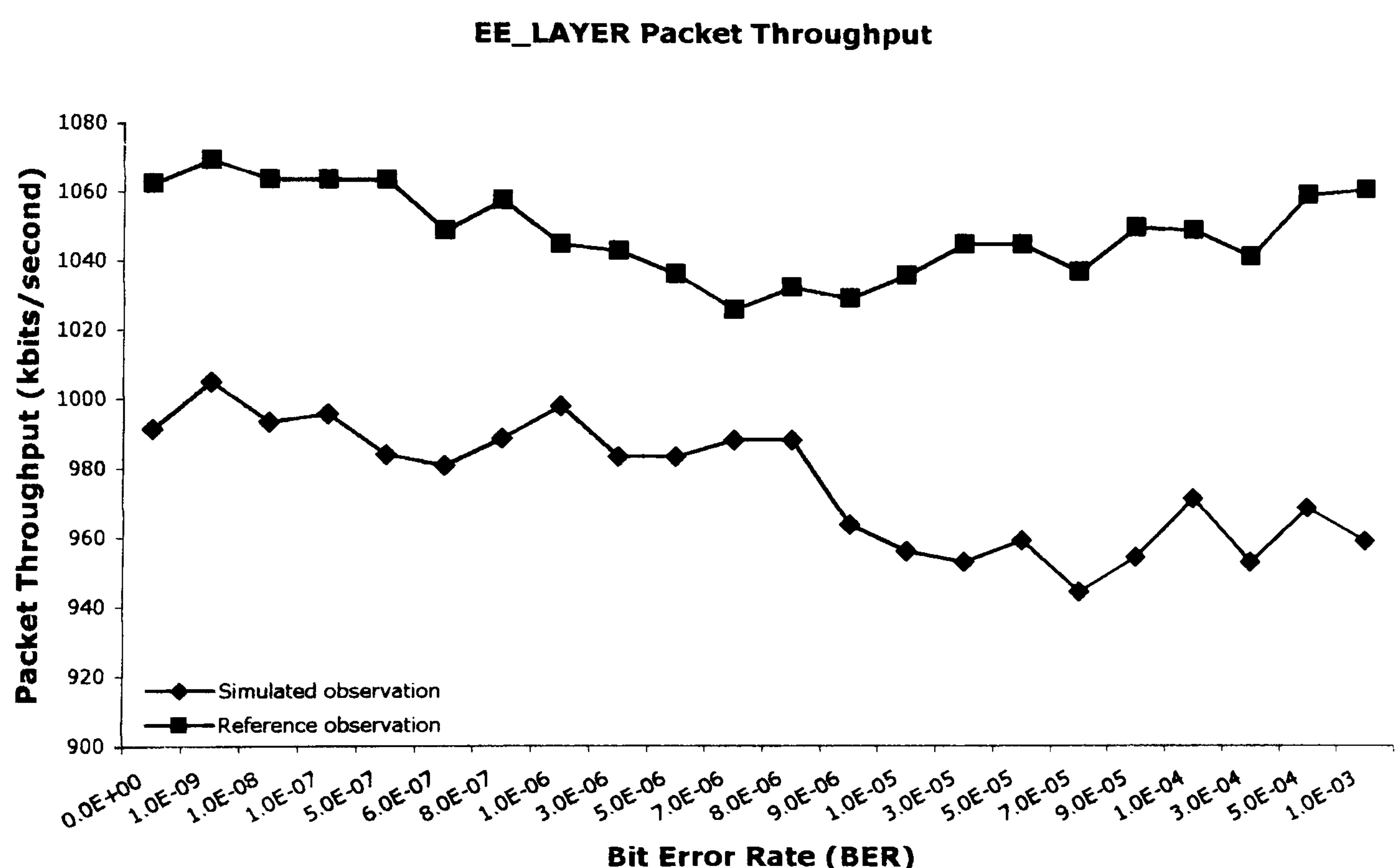


Figure 8.29: EE_LAYER packet throughput – RTT of 260 msec and variable BER

The packet throughput response of the RTT 260 msec topology conforms to the theoretical expectations, exhibiting a reduced performance due to the additional propagation delay imposed in relation to the reference measurement. The

throughput difference in the observations falls into the area of *64,133 bits* or *5 packets* for small error probabilities, rising to some *103,807 bits* or *8 packets* for larger BER. Due to the applicability notions of the OMNET++ and the adaptability of the FCNS in various network conditions, the response observed in Figure 8.29 can be accepted as the representation of the FCNS performance in delivering the user data packets for RTT of 260 msec.

8.3.2.2.6 RTT 260 msec – Frame Throughput

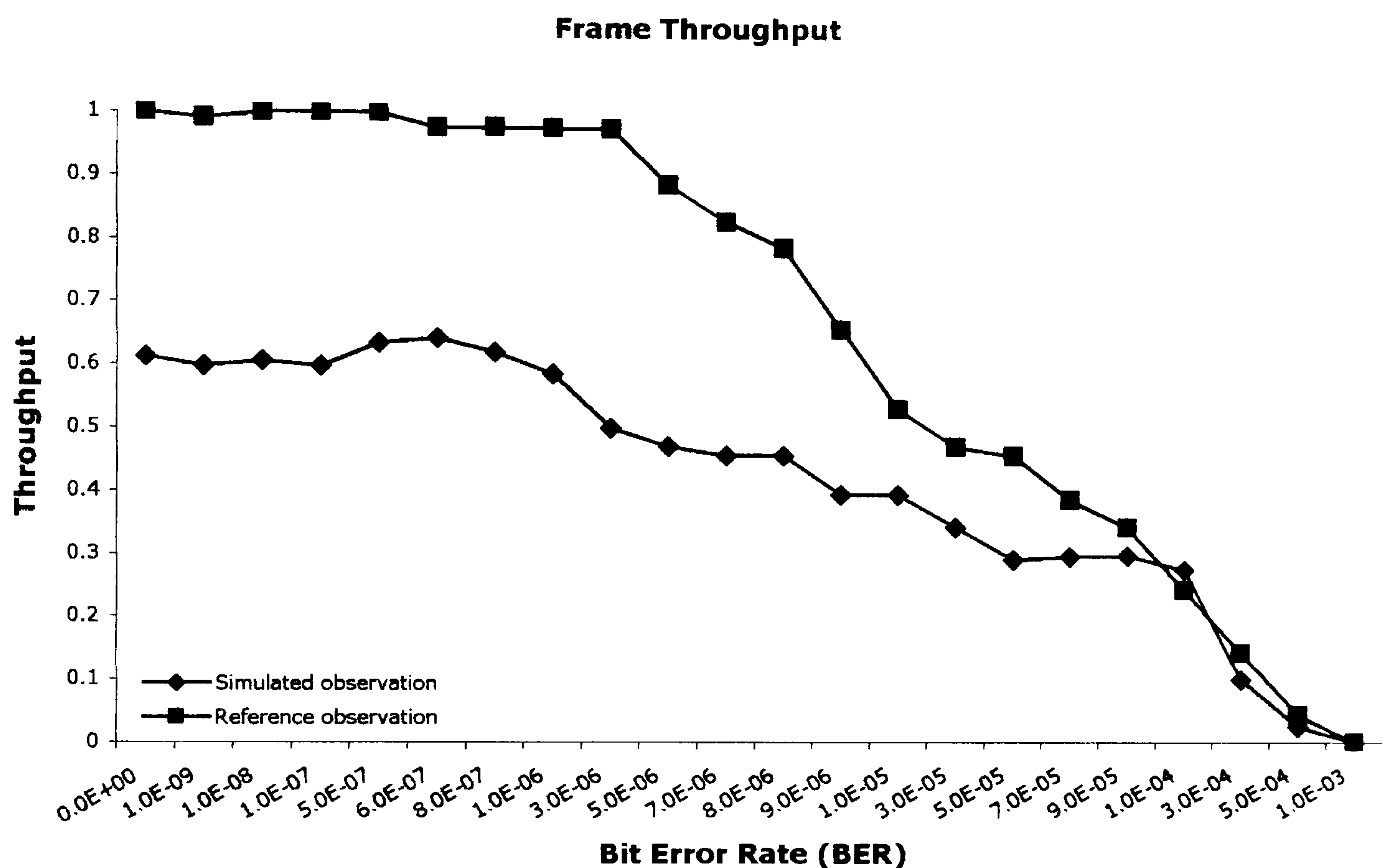


Figure 8.30: Frame throughput – RTT of 260 msec and variable BER

The deliverability ratio of Figure 8.30 is an illustration of the effects of the increased RTT delay in the PS network environment. Although the FCNS has proven to be adaptable in erroneous and delay-sensitive network conditions, high latency may lead to message losses, depending on the destination message timers. For this particular setting, timer expiration periods have been programmed to match the RTT to observe the response of the FCNS in such situation. Although these values are quite large compared to the specification settings of the FCNS (25-100 msec timers), the work has addressed the need of simulating cases where the FCNSEP

peer error signalling has been imperative. This has implied the loss of user data frames due to the FCNSEP message-processing overhead imposed in the network.

Even if the frame ACKs are piggybacked onto the ERROR_REP and ERROR_RESP messages, the receiving FCNS instance will have to process the FCNSEP message, consult the SL and/or the system management for the appropriate solution and transmit the response back the sender. Therefore, the overall loss induced to the system, even for an errorless transmission is not negligible, as is indicated in Figure 8.30. The work has therefore accepted the observation as valid for the RTT of 260 msec simulation setting.

The provision of both simulation environments has enabled the verification of the FCNS specification and the monitoring of the FCNS procedures and functions. The results presented are representative of the initial development of the FCNS and the refinement of the protocol services to provide a reference architecture for PS networks. To address the need of comparing the proposal presented in this thesis with currently used network protocols, the following section is provided, illustrating the performance of the FCNS in relation to the TCP/IP suite.

8.3.2.3 FCNS and the Internet Protocol Suite

In this section, comparisons between the FCNS architecture and the Internet suite are given, in relation to the protocol efficiency and throughput performances. For these measurements, two settings have been taken into consideration for the TCP/IP suite. In the first case, IPv4 has been assumed, without the use of additional security mechanisms to ensure the protection of the transmitted messages. In contrast, for the second case, IPv6 with ESP tunnel mode protection has been considered, to enable the evaluation of the FCNS against a secured network protocol architecture. Tables 8.1 to 8.3 depict the message header sizes for the respective protocol environments.

8.3.2.3.1 Relative Protocol Efficiency Comparisons – Variable Delay

Figures 8.31 and 8.32 represent the relative stack efficiency of the two protocol architectures for a constant data size of *1460 bytes* and variable BER. The graphs include comparisons with the IPv4 and IPv6 protocols respectively in relation to the FCNS when either full security measures are applied and with lack of security functionality.

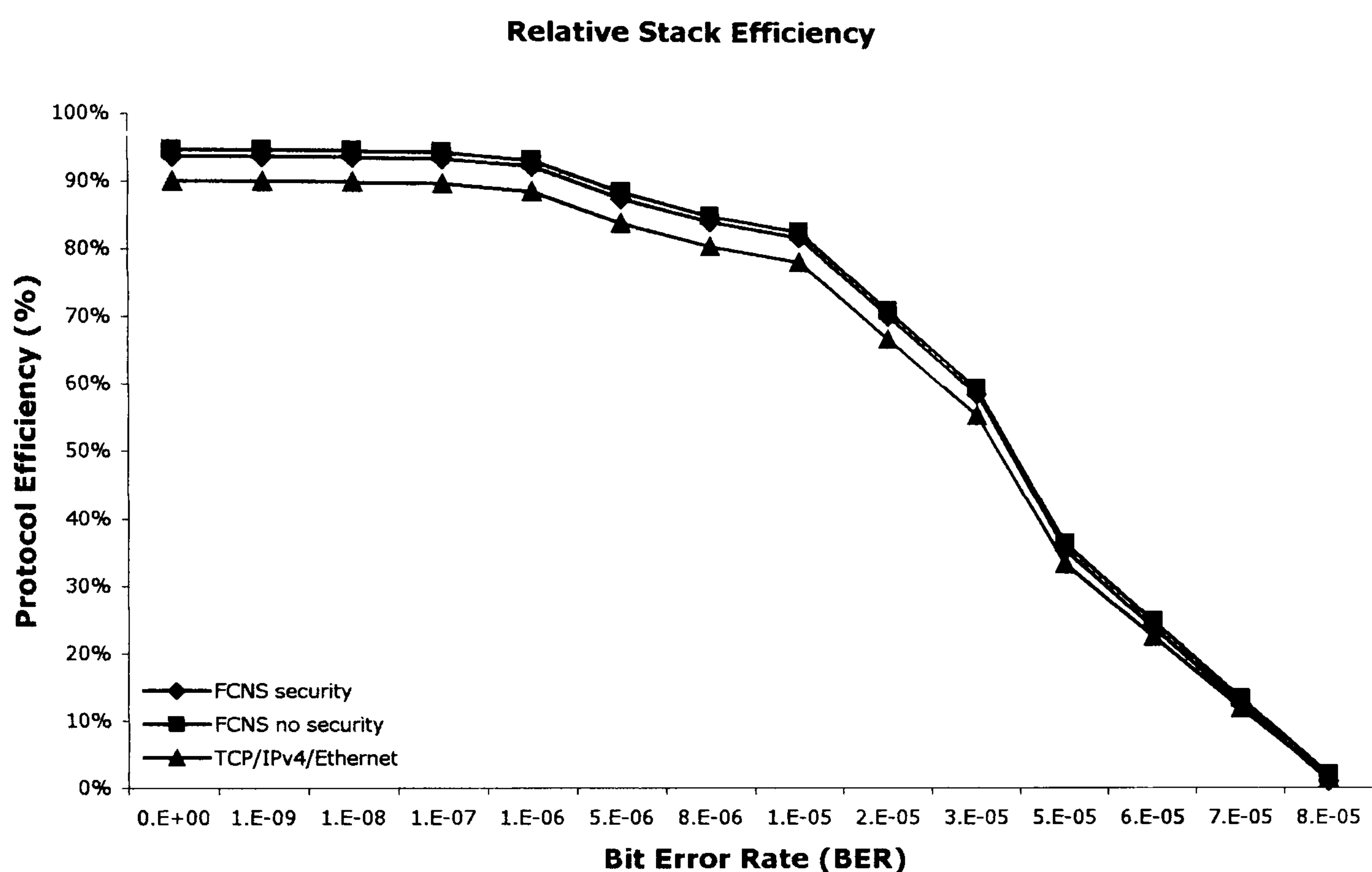


Figure 8.31: FCNS vs. TCP/IPv4 – Constant data size 1460 bytes and variable BER

The efficiency of the FCNS in delivering user data information is superior to that of the TCP/IP suite by a factor of 3.7% for the FCNS when full security measures are applied and of 4.61% when no security services are provided. Given that the 1460 data bytes constitute the optimum amount of data that can be transferred before fragmentation is required [14], it can be concluded that the FCNS can provide for a more efficient data transmission service according to the standards currently governing network operation. Even when full security measures are applied, the FCNS appears to exhibit greater performance than the TCP/IP suite, which for the particular measurements has been assumed to lack any security functionality. This fact strengthens the arguments presented in this thesis concerning the usage of

this proposal and the migration from currently used packet-switched protocol architectures to the FCNS.

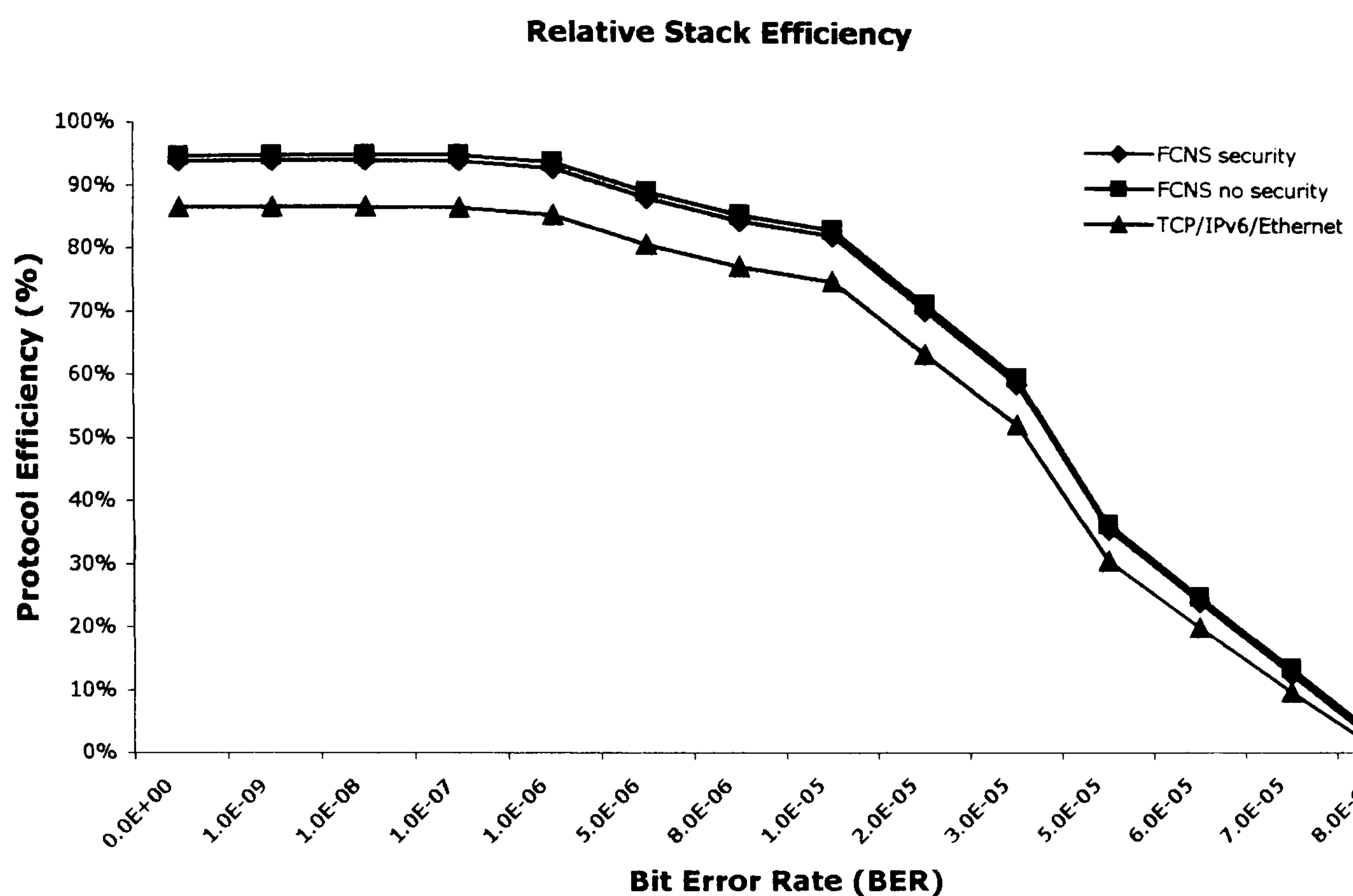


Figure 8.32: FCNS vs. TCP/IPv6 – Constant data size 1460 bytes and variable BER

Implementing into these calculations the newly adopted IPv6 standard, the work has assumed a level of security for the Internet packets, based on the application of the ESP in tunnel mode. The particular setting has been chosen, since ESP should provide for both packet authentication and integrity, in contrast to the AH that supports only the message authentication process. Therefore, the particular setting can be regarded as a possible real network backbone, enabling the comparison of the FCNS with a future network implementation.

As can be observed from Figure 8.32, FCNS achieves greater performance than the Internet suite, in the range of 7.32% for full security services and 8.23% with the FCNS lacking security functionality. Further advantages of FCNS include the support of security services choice by network operators at any layer of the protocol architecture, in contrast to the TCP/IPv6 suite, which supports only packet-level based protection mechanisms. The FCNS therefore outperforms the protocol

structure chosen to support the new generation of packet-switched networks, such as the UMTS CN.

8.3.2.3.2 Relative Protocol Efficiency Comparisons – Variable Data Size

Figures 8.33 and 8.34 illustrate the FCNS relative protocol efficiency with respect to the Internet suite for variable data sizes. Figure 8.33 depicts the results obtained for a representative error-free interval of 125 msec, whilst Figure 8.34 for a BER of 10^{-5} .

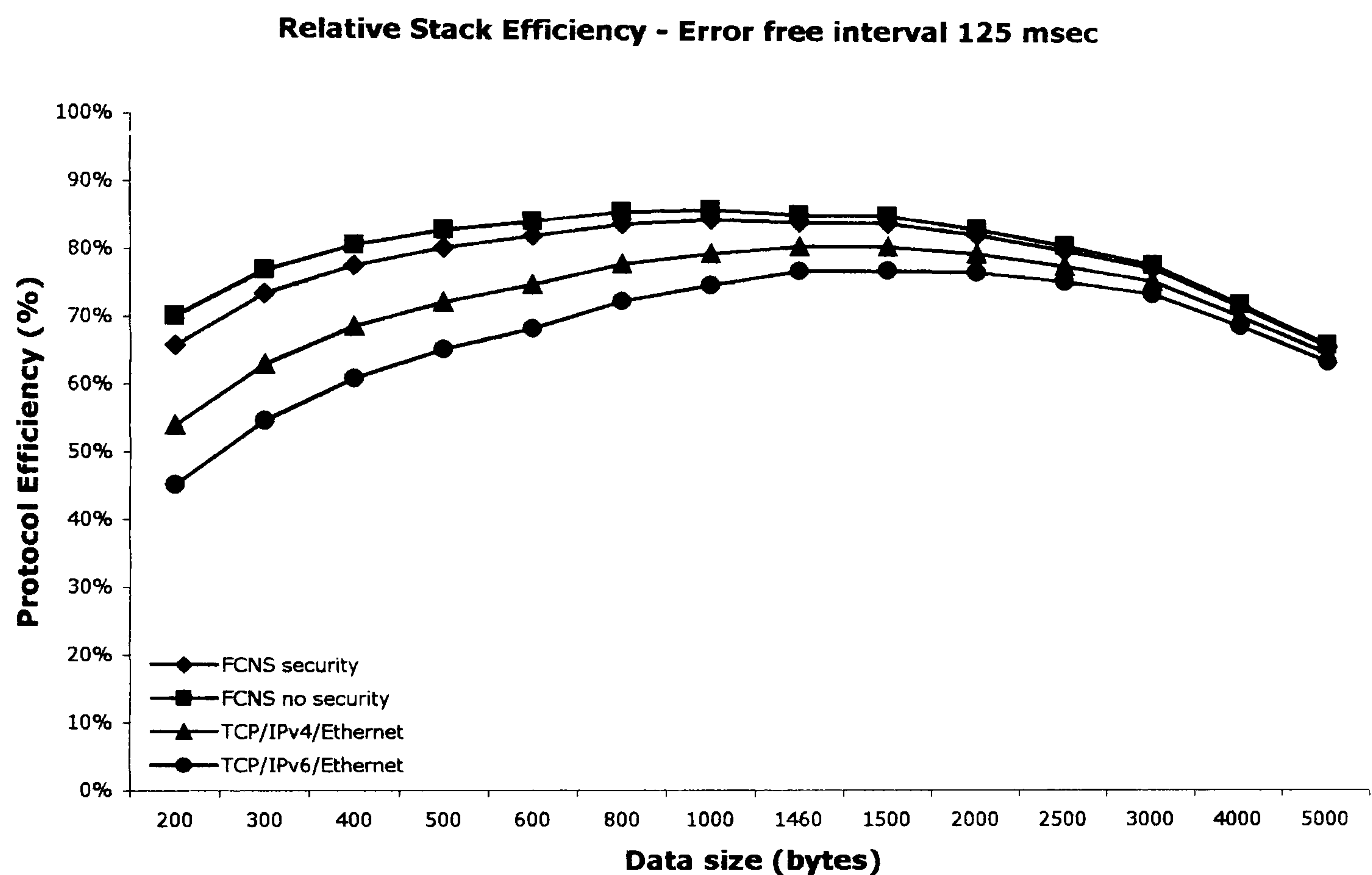


Figure 8.33: FCNS vs. TCP/IP – Constant BER (8×10^{-6}) and variable data size

The measurements presented in Figure 8.33 illustrate attempts made to identify the performance of the FCNS in relation to the Internet suite for various data sizes. As can be seen from the depicted response, FCNS has a 12% superior performance to the TCP/IPv4 suite and an increased efficiency over a secured Internet topology in the area of 21%. In case no security measures are applied, then the FCNS outperforms the Internet suite by 16.2% for the IPv4 case and by 25% when IPv6 is used. The increased FCNS efficiency in relation to TCP/IP can be observed for small data sizes, typical of the signalling data that may be transferred in a telecommunications network.

To further enable the comparison between the two protocol suites, the following results depict the FCNS superior efficiency for an error free interval of 100 msec.

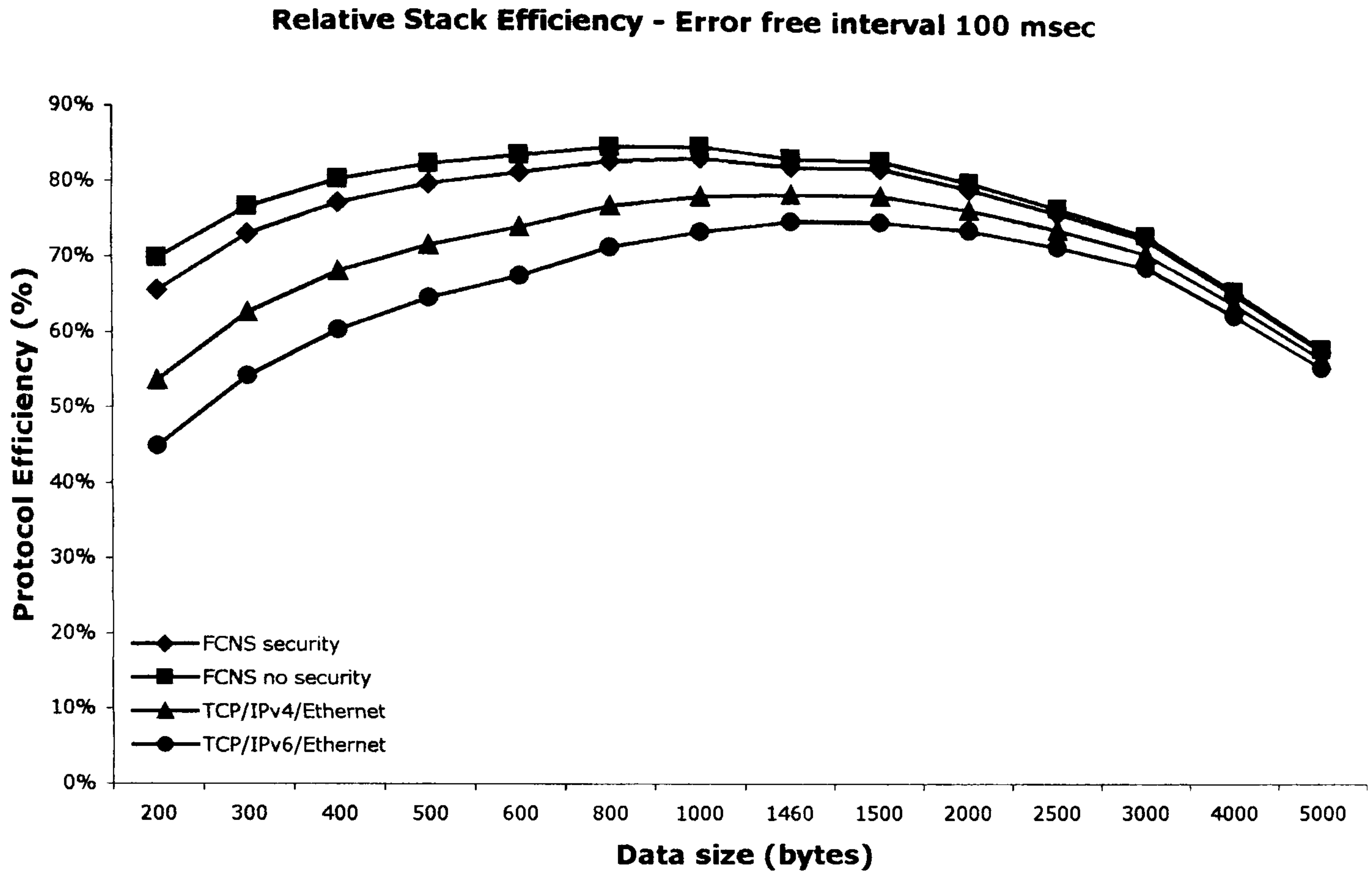


Figure 8.34: FCNS vs. TCP/IP – Constant BER (10^{-5}) and variable data size

For small data sizes, FCNS with full security measures presents greater performance of 12% in relation to the TCP/IPv4 suite and 15.5% from the TCP/IPv6. Lacking security functionality, the FCNS is superior by 16.2% and 20% respectively.

For both error-free interval cases, as the data size increases, FCNS still exhibits greater performance than the Internet suite, supporting the arguments that it can be better suited for packet data transmission in PS environments. The application of the security measures does not affect the efficiency of the protocol stack to a great extend. Consequently, network operators can choose to realise the full protection mechanisms for the FCNS architecture without losing functionality due to the increased message sizes and functionality supporting the security functions of the stack.

The following subsections depict the throughput performance of both suites, in an attempt to provide the reader the means of identifying the performance of the FCNS to deliver the requested data, in relation to the TCP/IP protocol suite.

8.3.2.3.3 Throughput Comparisons – Acknowledging all messages

For these sets of measurements, the work has assumed that for the FCNS architecture all messages should be acknowledged prior to the transmission of the next frame. For the TCP/IP suite, ACK has been requested for every 2 messages, to enable the support for the lowest possible receiver window size of TCP/IP [14].

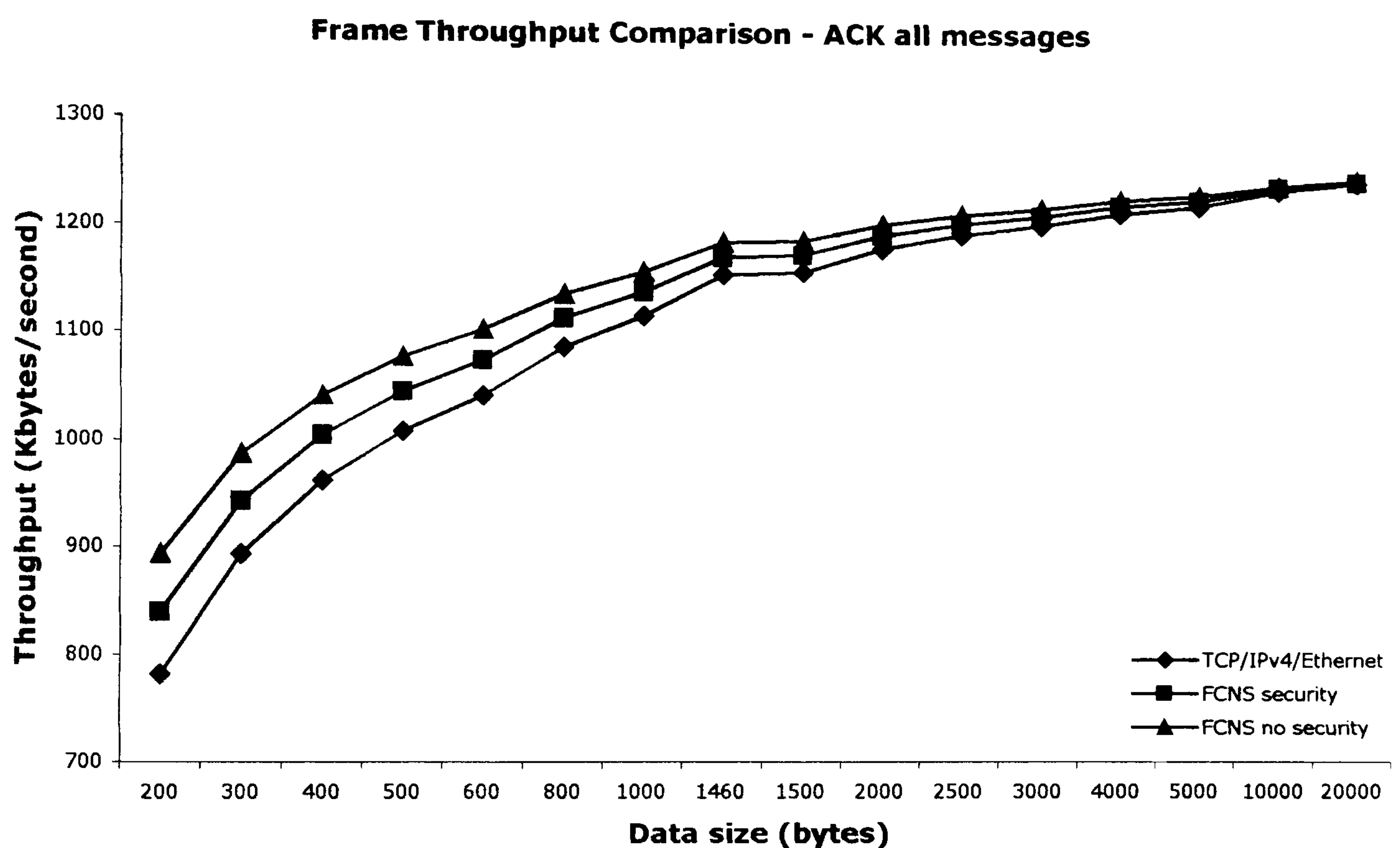


Figure 8.35: FCNS vs. TCP/IPv4 – Variable data size throughput responses

The observations depicted in Figure 8.35 represent an errorless transmission, to enable the provision of the highest theoretical throughput that both suites can achieve. The datarate taken into consideration has been chosen to be 10 Mbps, to support consistency with the rest of the measurements given in this chapter.

FCNS has a superior performance over the TCP/IPv4 architecture in the order of *111,607 bytes* and *52,676 bytes* depending on whether no security functionality has been enabled or when full security measures are applied. In terms of the

additional amount of the user data that can be transferred, then for a representative data size of 1460 bytes, FCNS can support the transmission of 20 more packets when no security measures are applied and 19 more packets with full security functionality. In the context of a data transfer process where every message should be acknowledged, the advantages and superior efficiency of the FCNS stack become apparent.

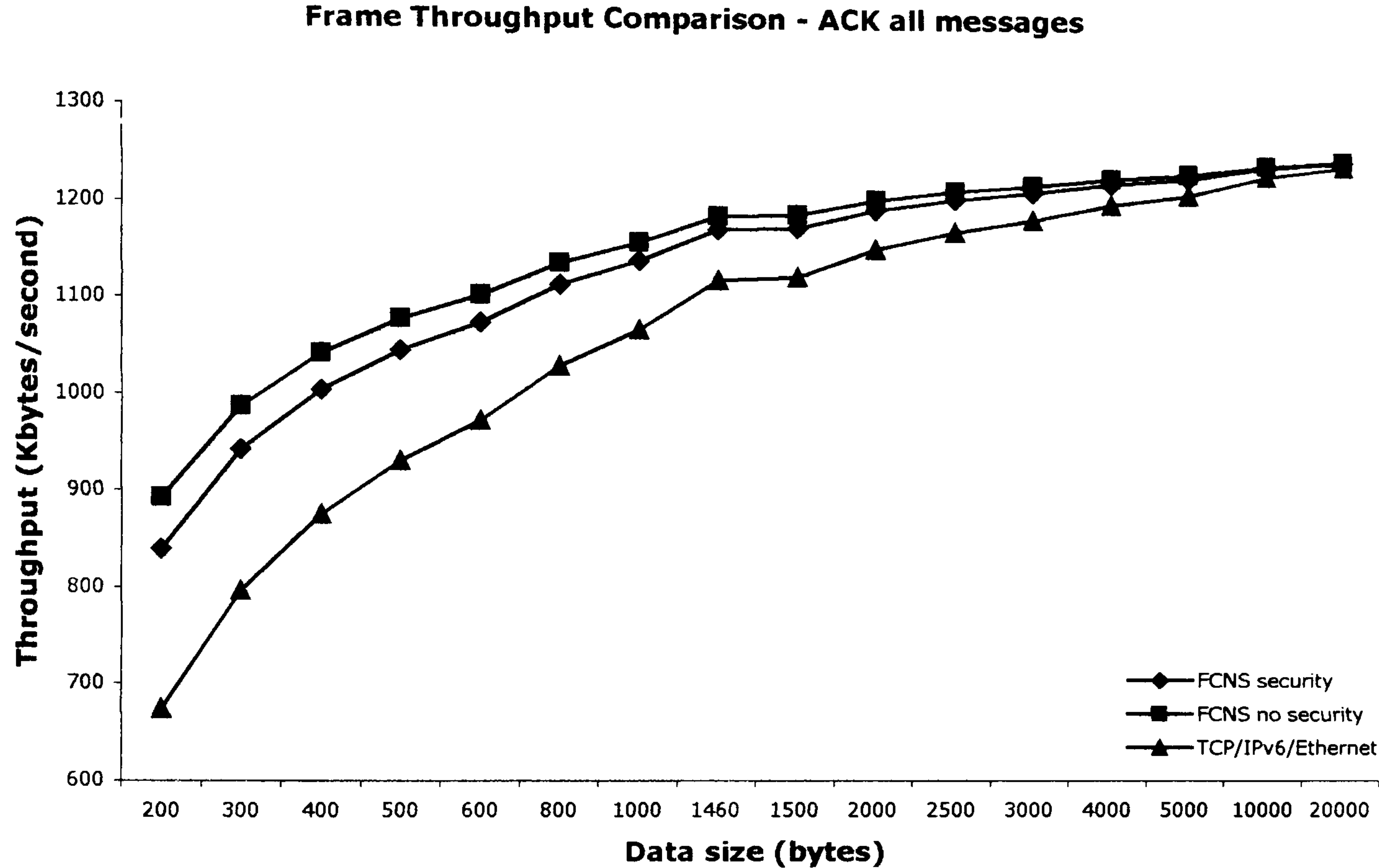


Figure 8.36: FCNS vs. TCP/IPv6 – Variable data size throughput responses

FCNS exhibits greater throughput efficiency in relation to the TCP/IPv6 suite as is presented in Figure 8.36. Due to the increased header sizes compared to the TCP/IPv4, the Internet suite displays an inferior to the FCNS performance in delivering user and/or signalling data between the network elements. Given that FCNS can transfer more messages for the same data size, it can be concluded that the communication channels will be better utilised, resulting in a more efficient data transmission.

8.3.2.3.4 Throughput Comparisons – Acknowledging message blocks

The superiority of the FCNS can also be illustrated for the maximum receiver window sizes set for the two architectures. For the FCNS stack, an

acknowledgement is requested every 32 messages, whilst for the TCP/IP an ACK is required every 22 frames. The latter is due to the 65535 - window size of the TCP receiver (64kbytes of application data for the 16-bit window size of the TCP header), which for 1460 bytes of user data results in 44 messages being transmitted. Since messages have to be acknowledged, receiver sends an ACK every 22 messages to support the continuation of the data transmission process and the avoidance of retransmitting a whole block in case of bit errors. It should be noted that window sizes greater than 64 kilobytes can be supported [186], yet, this still remains on a theoretical level.

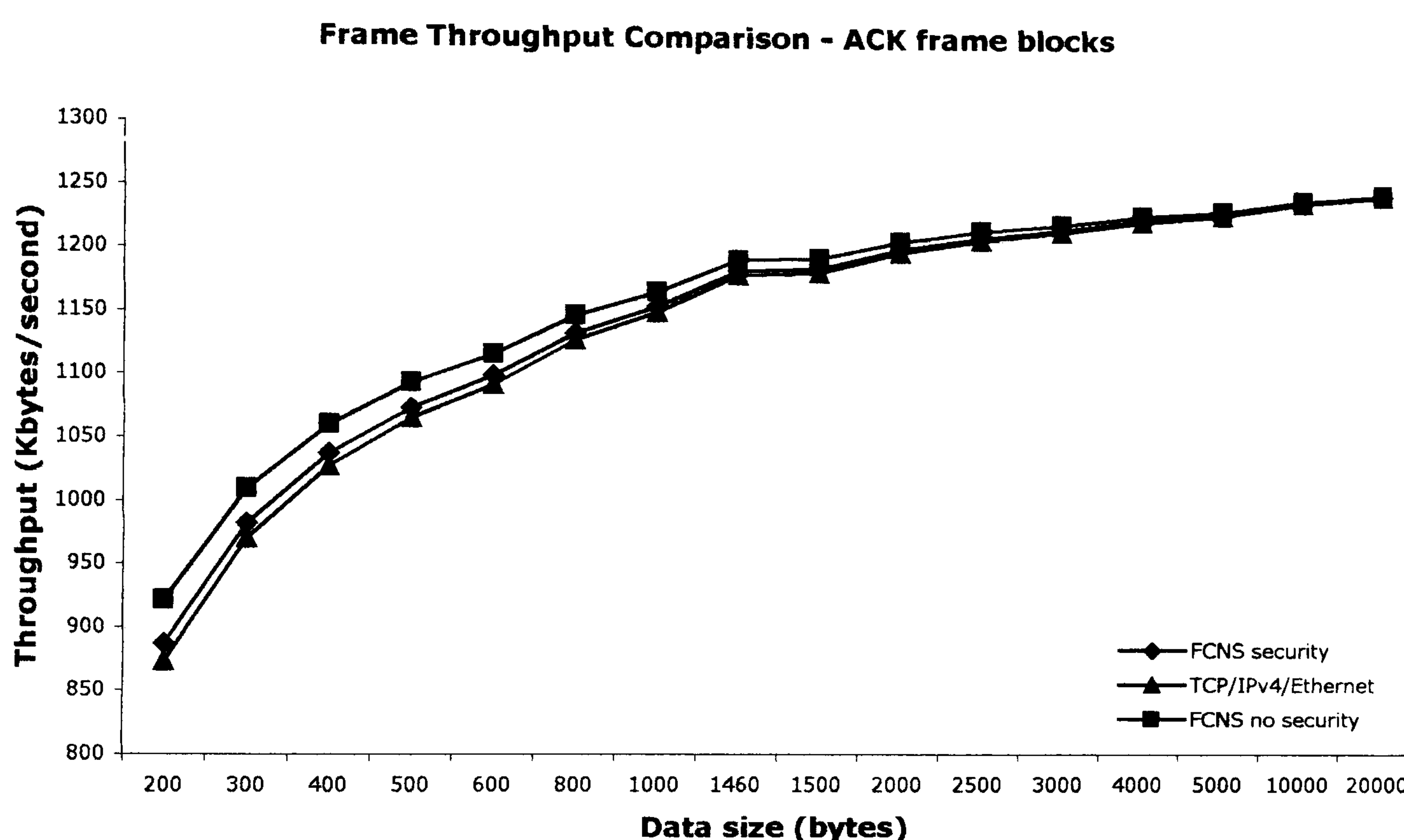


Figure 8.37: FCNS vs. TCP/IPv4 – Variable data size throughput responses acknowledging frame blocks

The performance of the FCNS appears to be greater than that of TCP/IPv4, in the range of 32,925 to 48,572 bytes for small data sizes, that is, 21 to 32 messages. This implies that an additional message block can be transmitted with the FCNS resulting in a more efficient and quicker data transfer process.

FCNS also outperforms the TCP/IPv6 suite for the case of an errorless transmission with varying types of user and signalling data. The amount of additional frames that could be sent using the FCNS at similar transmission times varies from 15 to

55 when full security measures are applied and from 23 to 87 for the FCNS with no security functionality.

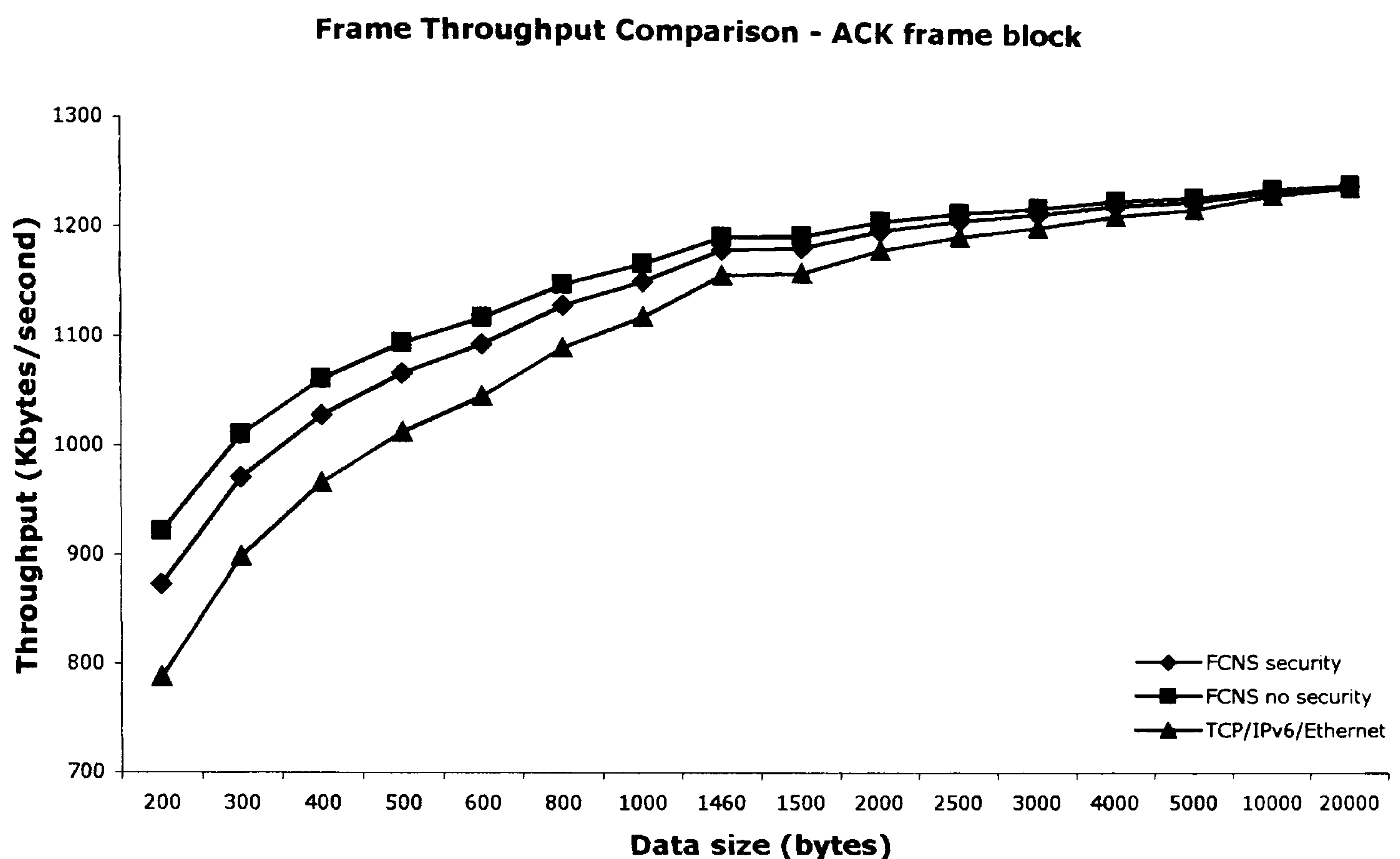


Figure 8.38: FCNS vs. TCP/IPv6 - Variable data size throughput responses acknowledging frame blocks

Finally, Figure 8.39 depicts the throughput efficiency of FCNS in relation to the TCP/IPv6 suite for variable BER and data rate of 1.5 Mbps. The throughput of the FCNS and TCP/IP stacks has been considered, for a 125 msec constant error-free interval and data size of 1460 bytes. Even with the application of the maximum possible security measures for the FCNS communication layers, FCNS achieves much greater throughput than the TCP/IPv6 suite, particularly in adverse BER channels. In all cases, FCNS is 10% superior to TCP/IP, rising to some 30% at high BERs.

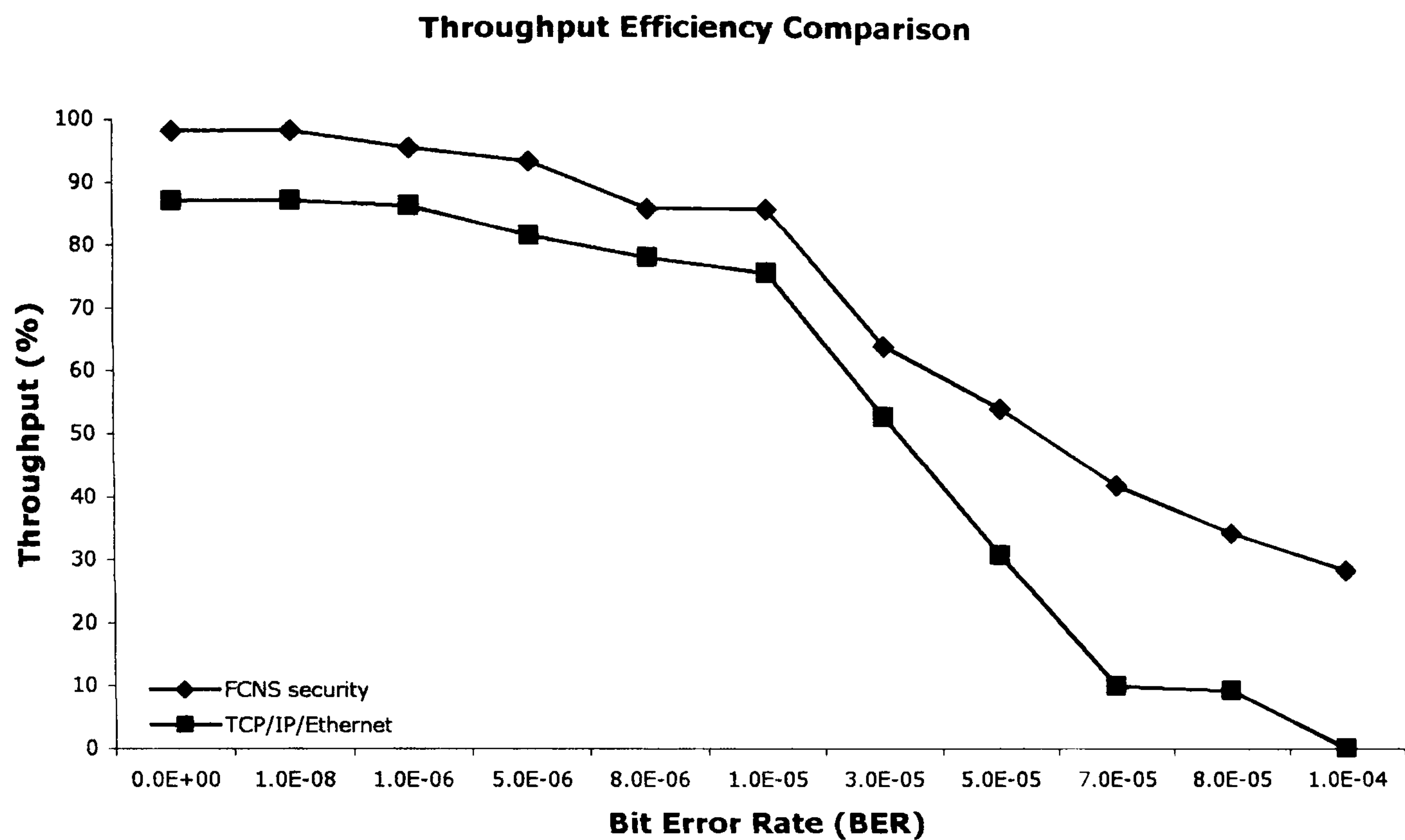


Figure 8.39: FCNS vs. TCP/IPv6 – throughput efficiency comparison

From the measurements presented in the preceding section, there can be the conclusion that FCNS is more suited to serve as the means of transporting user and signalling information in telecommunication systems. Chapter 9 details the applicability issues of the FCNS for the UMTS CN.

Chapter 9: Conclusions & Suggestions for Future Work

Chapter 9. Conclusions and Suggestions for Future Work

9.1 Overview

9.2 FCNS Applicability

9.3 Further Work

In Chapter 9 the conclusions drawn from the research work are presented. The major arguments and points made for the FCNS stack are given followed by an insight on the applicability of the proposed protocol for telecommunication systems. Finally, further work is proposed to enhance the FCNS capabilities and usage in future telecommunication networks.

9.1 Overview

The focus of this thesis has been the design and development of a secure protocol architectural framework for use within packet-switched network environments. The provision of a stack that is independent from currently used structures enabled the design of a reference model addressing the issues of message protection for both the internal and external protocol data units.

The research work has presented a novel protocol architecture, realising security services to all level of the communication between the stack layers. Protection of the layer service primitives and PDUs exchanged between the layer interfaces are issues that to our knowledge have never arisen in previous protocol designs. The increased probability of attacks launched against messages in transit, leveraged the design of communication protocols in which security architectures have been added to support the required functionality.

The idea behind the design of the FCNS has been the need of a scheme that would firstly protect the internal messages of a set of communication rules, before proceeding in providing services for the actual information to be transferred. The reasons for this approach lay on the technology supporting network connectivity and the integration of communication systems into the UMTS. The demystification of telecommunication networks implied the usage of protocols widely used and exploited in computer systems. The technological advances in cryptographic applications meant the provision of more secure algorithms, minimising the possibilities of successful attacks. It has therefore been identified that attempts at manipulating a connection would target the operation of the protocols supporting the communication, since no mechanisms have been used to account for their protection.

The FCNS structure has been viewed with respect to the individuality each layer exhibited, allowing the security of the protocol modules irrespective of the status of

the whole stack. The Security Layer has been designed to provide each FCNS layer with the necessary functionality to ensure the protection of the communication process at all possible levels. All messages concerning the operation of the FCNS have been secured, to counter possible attacks aiming at connection discontinuity, information disclosure, session replay and hijacking, as well as illicit connection requests.

The security functionality of the FCNS has been encompassed into a single layer, to reduce problems identified with reference architectures, such as the OSI model. The lack of implicit mechanisms in the FCNS communication layers signified the avoidance of functionality duplication and, at the same time, the ease in managing and updating functions in relation to network security advances. The SL has consequently been able to support a wide range of security services to all levels of communication, reducing the processing load and network resources that would be required for each individual layer.

The advantage of the FCNS architecture is the simplicity of the design of the stack, offering network operators the possibility of enforcing security mechanisms on-demand and at any required level. The application of the SL protection mechanisms is not imperative, enhancing the flexibility of the FCNS in transmitting various types of data.

The FCNS conforms to the design principles of the OSI model, adopting a layered structure to maintain compatibility with currently used protocol architectures. The most distinctive features of the research work proposal are the lack of clearly defined application and physical layers, as well as the placement of the SL in relation to the rest of the stack. The exclusion of the layered functionality for the mentioned protocols is explained in Chapter 4, where an indication is given as to the steps taken to provide similar instances for the FCNS simulation environments. The layers analysis has proven that the FCNS communication layers can adapt to

many network environments, offering network operators the choice of either using the FCNS stack as is, or replacing its layers with other protocols.

The work has also analysed the design and applicability of the FCNS keystream generator, serving as the provider of the secret keys used for the security contexts (SCs) of the internal messages. The FCNS external messages can be protected by using either the FCNS keystream generator or parameters provided by the network, such as the HLR/AuC of the UMTS CN. The generator has been subjected to tests regarding its applicability in security applications, referencing standardised procedures to provide a system that could be used in current topologies. Further statistical and cryptographic tests have also been used, to identify the generator correctness and relevance with respect to the current status of cryptographic applications. The FCNS keystream generator has successfully passed the tests in question, making it suitable for the provision of the necessary secret keys for the FCNS SCs.

Furthermore, the FCNSEP has been presented, providing details on the operation of the protocol and the application of security functions for its messages. FCNSEP can be used for either interlayer or peer-error signalling, to indicate a wide range of faulty conditions and obtain the appropriate actions by the SL or the system management. The protection of the FCNSEP messages depends on the level of its application and signalling procedures. For this reason, the SL has to exchange the respective SCs with the FCNS communication layers, to enable identification of the signalling case and the provision of the required security services. The FCNSEP has been compared to currently used signalling techniques, such as the ICMP of the TCP/IP suite. The analysis of the ICMP has proven that FCNSEP provides a more complete and efficient solution in notifying the protocol layers and the peer communicating entities of transmission errors and fault conditions that might occur. In particular, the independence of the FCNSEP from specific communication protocols implies that any layer can signal a condition to the SL or the network

system. By this method, errors regarding the negotiation of the transfer syntax, QoS parameters and SCs can be indicated, in conjunction to the cases arisen from congestion built up and unreachable hosts.

The FCNS prototype has also been developed, which addressed the issues of verifying the communication between the FCNS layers and the detection of potential design pitfalls. The abstraction of the FCNS specification enabled also the comparison of the FCNS stack to the design expectations and its conformance to the procedure rules set for the services it offers to network elements. The logical consistency of the specification itself has also been tested, to detect any functionality duplication and the definition of functions and services outside the scope of the protocol.

Following the validation of the FCNS prototype, the design has been realised into software code. This approach has been followed to create the FCNS simulation topologies, for the testing of the FCNS functions and procedures in relation to connection requests and data transfer processes. In particular, the software model enabled the observation of the effects of the security functions to the overall communication process, both at the internal and external levels. Since emphasis has been given on the security of the internal FCNS messages, the layers of the protocol structure have been viewed as independent modules, connected to each other via links. These links actually represented the interfaces of a layered protocol to those directly adjacent to it, accepting and forwarding information to the access points specific for a service. Their use provided the advantage of programming the delay it would take for a layer to process a message, simulating the effects of the FCNS functions application.

The model enabled the comparison with currently used protocol architectures and, more specifically, the TCP/IP. These measurements targeted the applicability of the architectures in supporting the communication between the UMTS CN elements. Of

importance, observations related to the efficiency and throughput of the suites in delivering signalling type data, for cases where security functionality was not needed and when full security measures needed to be applied, have been taken. FCNS outperformed the Internet suite for the cases included in the observations, varying from the application of variable data sizes to BER alterations.

For the protocol efficiency calculations, the FCNS exhibited 3.7% greater performance than the TCP/IP suite running with IPv4 and 7.32% for the IPv6 with ESP in tunnel mode, for a representative 1460 byte long data size and variable BER. Varying the data size to reflect several cases and types of information exchange, then for an error-free interval of 125 msec, FCNS had a superior performance of 12% in relation to TCP/IPv4 and 21% to TCP/IPv6. Increasing the BER to 10^{-5} , FCNS achieved 12% greater performance with respect to TCP/IPv4 and 15.5% to TCP/IPv6. The values indicated, reflected observations made for the FCNS when full security measures were applied, to model a reference FCNS instance.

For the throughput calculations, FCNS has exhibited superior functioning compared to the Internet suite. For both cases of a minimum and maximum receiver size window, FCNS achieved greater throughput performance than TCP/IP, varying from an overall transmission of 19 to 55 additional frames respectively.

Finally, comparisons have been made to observe the protocol throughput efficiency under BER variations and constant data size. FCNS has indicated superior performance to the Internet protocol architecture, of the order of 10% to 30%, even for relatively small error-free intervals.

There has therefore been the conclusion that the FCNS exhibits superior performance to the TCP/IP suite under real network conditions, involving the transmission of various sized data in erroneous communication channels.

9.2 FCNS Applicability

In this section an insight on the UMTS CN services is given, to enable the identification of the FCNS applicability in telecommunication systems. The architecture chosen to attempt and analyse the FCNS capabilities is the Release 99 of the UMTS CN and more specifically the PS domain. Figure A.1 of Appendix A illustrates the view of the R99 network as developed in the 3GPP specifications [3].

The PS domain forms the set of the CN entities offering a PS type of connection for both the user traffic and network signalling [187]. User information is transferred via data packets each of which can be routed independently of the previous one. In a sense, the PS domain exhibits the characteristics of a general packet-based network, such as the Internet in the way messages are exchanged between the network nodes.

The elements of interest to the PS domain are the *Equipment Identity Register (EIR)*, the *Home Location Register (HLR)* encompassing the *Authentication Centre (AuC)*, the *3G Serving GPRS Support Node (SGSN)* and the *Gateway GPRS Support Node (GGSN)*. The UMTS registers are actually independent of the domain CN refers to, although their inclusion is imperative due to the information they provide to the PS network elements.

The EIR contains information regarding the user identity and mainly the equipment numbers that will be used throughout the connection. The HLR/AuC contains a collection of information to which a subscriber is assigned for record purposes. The AuC stores data for each user to enable its authentication whenever a connection needs to be established. The SGSN is a register used to support subscription information for PS services and the communication of the CN towards the access network. It is responsible for mobility management issues such as routing area and location updates, as well as controlling the security mechanisms relevant to the packet connections. Finally, the GGSN maintains the connections towards external

packet-switched networks, such as the Internet, and has the session management responsibilities of the UMTS CN. It is also used to store subscriber data received from the HLR and/or SGSN.

The four basic QoS classes that must be supported in the UMTS CN are as follows [65]:

- Conversational class: minimum fixed delay, no buffering, symmetric traffic, guaranteed bit rate. This class is usually applied for time sensitive data such as real-time audio and video.
- Streaming class: minimum variable delay, buffering allowed, asymmetric traffic, guaranteed bit rate. Applied mainly for streaming applications such as music downloading.
- Interactive class: moderate variable delay, buffering allowed, asymmetric traffic, no guaranteed bit rate. An example of such service is the Wireless Application Protocol (WAP).
- Background class: big variable delay, buffering allowed, asymmetric traffic, no guaranteed bit rate. Support for general packet-based transmissions.

The guaranteed bit rate parameter is the rate that the bearer service must carry between the service end points, given the maximum permissible rate that may be used. The identification of the service classes enable the designer in identifying the applicability of the set of communication rules that can be used to support the required services.

FCNS supports many levels of QoS for a packet-based data transmission. Depending on the data that has to be exchanged, the various classes can be supported by the UDSES and TX_LAYER, giving priority to time and delay sensitive data. At the same time, the full – duplex communication support and FCNS flow control mechanisms enable the provision of either symmetric or asymmetric traffic.

The FCNS ability to adapt to erroneous conditions, such as the presence of excessive network latency and bit errors, makes the protocol stack an attractive solution for supporting the four service classes of the UMTS CN and consequently the transmission of virtually any type of traffic.

For the PDP context exchanges, the observations for the propagation delay of 100 msec have provided the means of extracting information as to the efficiency of the FCNS in handling signalling-type information data requests. The characteristics and particularities of this kind of data have been modelled by setting the data size in 200 bytes or 1600 bits and by setting the priority of the data to the highest possible. From the results obtained, the work has concluded that FCNS can efficiently support the basic UMTS services, due to the protocol's high adaptability and flexibility in transmitting signalling type data.

For the UMTS CN, security is also a very important QoS factor as analysed in Chapters 2 and 3. For the R99, there exists no network domain security defined for the PS domain. Because R99 will be the first commercial UMTS release, it is very important for the operators to be able to support an adequate level of protection for the sensitive data that may be transferred between the network elements.

FCNS addresses the issues of message security irrespective of the underlying submedium and the user data. This transparency enables the protection of the user and/or signalling data at the highest possible level, assuming that there exists an adequate mechanism supporting the creation and exchange of public keys used between the peer elements.

The provision of the FCNS traffic padding mechanism ensures that the possibilities of a successful passive monitoring attack (sniffing) are minimised, avoiding traffic-pattern attacks. By this method, sensitive data such as secret keys used for the

user authentication procedures cannot be distinguished from random data, disabling an attacker from manipulating the information in transit.

Furthermore, the FCNS keystream generator provides for the unique keys used to secure the FCNS messages. The SC exchange process ensures the session integrity, making session hijacking attacks very difficult to succeed. This implies that valuable user information that may be exchanged between the SGSN and GGSN will not be disclosed to an unauthorised party and, at the same time, the adversary will not be able to repeat part or the whole of the session to gain information about the network and/or its subscribers.

The provision of the unique FCNS keys, the address verification procedures and the SL functionality also support the protection of the system against illicit connection requests that may lead to DoS attacks [65]. Also, protection is provided against spoofing attacks aiming at forging service requests. It should be very difficult for an adversary to masquerade a legitimate node and request the transmission or manipulation of user and/or signalling data.

The implementation of the FCNS can therefore provide for a more flexible and efficient solution for use in telecommunication environments than currently defined protocol architectures. The simplicity of the proposed architecture enables the portability of the FCNS in many packet-switched environments, and also the update of the protocol mechanisms depending on network and communication advances.

The superior performance of the FCNS in relation to the TCP/IP suite, as observed in Chapter 8, enhances the arguments presented in this thesis, for which FCNS should be the preferred solution for supporting packet-based communication. For the UMTS CN, there is also the advantage of preserving compatibility between the various architectural releases. Updates on the UMTS should reflect possible amendments to the FCNS functionality and not the choice of additional or different

protocols serving as the network backbone. Cost is therefore minimised by maintaining this level of consistency and at the same time provide for an easy to manage solution portable for any service users might request.

9.3 Further Work

The research work has presented a secure architectural framework for use within PS network topologies. The FCNS can be used in environments such as computer and telecommunication systems, addressing protocol and data security issues. The comparisons with the TCP/IP suite have indicated the superiority of the proposed work in providing a more efficient transmission for a wide range of data types. Its applicability has been analysed with respect to the UMTS CN, affording a sufficient and flexible solution for the security of the R99 architecture.

The FCNS specification represents the initial attempts made to define an up-to-date reference model. The simulation environments enabled the testing of the protocol functions and procedures, identifying effects such as security functions application and the FCNSEP. To further address the applicability of the FCNS in telecommunication networks, the model can be transferred into the OPNET modeller environment, which makes use of implicit libraries providing a more complete solution. Cases include the UMTS CN architectures, wireless networks, LANs and mobile access systems. The usage of such a commercial tool would enable a more pragmatic FCNS evaluation, since libraries are pre-programmed, offering greater flexibility in addressing design and development issues of the stack.

Furthermore, the FCNS can be extended to support variable bit rate applications such as multimedia. The measurements presented in Chapter 8 indicate a rate of adaptability of the protocol in BER and delay variations, which is of particular importance when transferring time and delay-sensitive data. Due to the nature of the proposal, multimedia support has not been investigated in depth and issues

such as the establishment and monitoring of video-conferencing connections would merit further study. Support for such services has only been viewed with respect to the implications security functionality might have, where the SL may not be needed for specific layers of the communication.

An additional feature that can be exploited is the transmission of voice data for the UMTS CS domain. Messages originating from the Mobile Application Part (MAP) protocol can be mapped onto the FCNS frames and be exchanged between the UMTS network elements. As the architectural releases of the UMTS progress, commercial deployment of the network will imply the support of a single domain, where voice and packet data will be transferred by a single protocol (e.g. voice over IP). To preserve compatibility with R99, given that the FCNS is used as the transport mechanism for the UMTS signalling data, voice-based services can be developed to adapt to the technological advances and nature of the integrated system.

Finally, the FCNS PHYS layer functionality can be enhanced to support network environments such as local area networks. The particularity of such topologies entails the classification of the network nodes into servers and clients, in which elements are not peers. The UDSES protocol already includes services such as the provision of communication tokens and their management. The PHYS layer though has not been designed to address communication procedures for LANs. The provision of a Medium Access Control (MAC) scheme and token technology can therefore be researched, to enable the use of the FCNS in such topologies. Due to the simplicity and architecture of the protocol stack, updates will only be necessary for the PHYS layer. The SL will be able to provide the appropriate SCs no matter the level of network support PHYS layer affords and the environment on which FCNS runs.

The FCNS has been designed to be simple and flexible in adapting to network and security technological advances, increasing its applicability in network environments. The development of the protocol stack should therefore be continuous, to preserve its efficiency and compatibility with standardised network models and transmission techniques. Its increased performance in relation to currently used protocol suites and the advantages it exhibits over standardised reference models, make the FCNS an attractive solution for use in current and future telecommunication networks.

Appendices

Appendix A. FCNS Protocol Specification Diagrams

Appendix A includes the abstracted representations of the FCNS layers processes, given in SDL diagrams. Information is also provided as to the UMTS CN R99 architecture.

Figure A.1 – UMTS CN R99

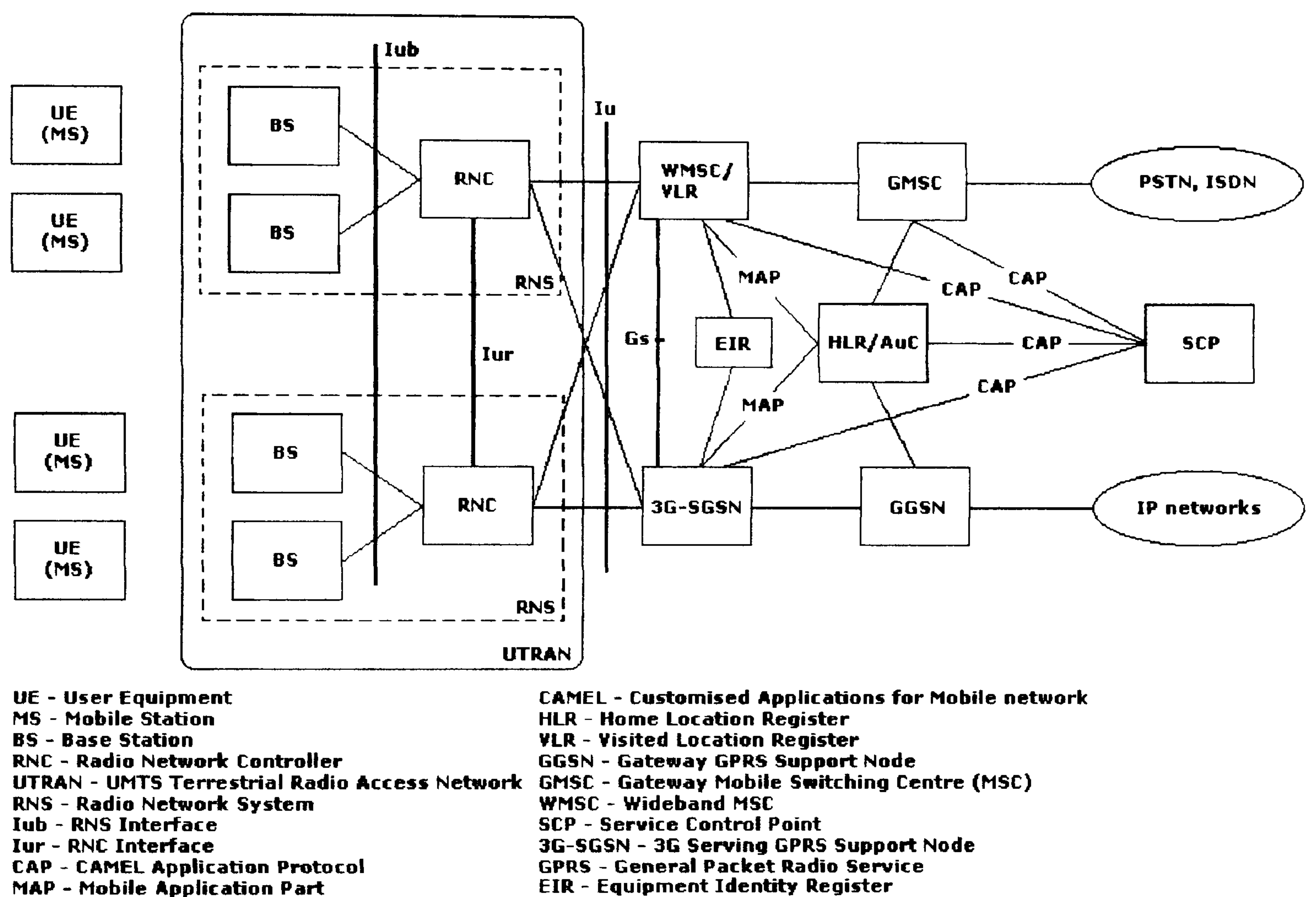


Figure A.2.1 – Node Sending Instance

Process Type sending

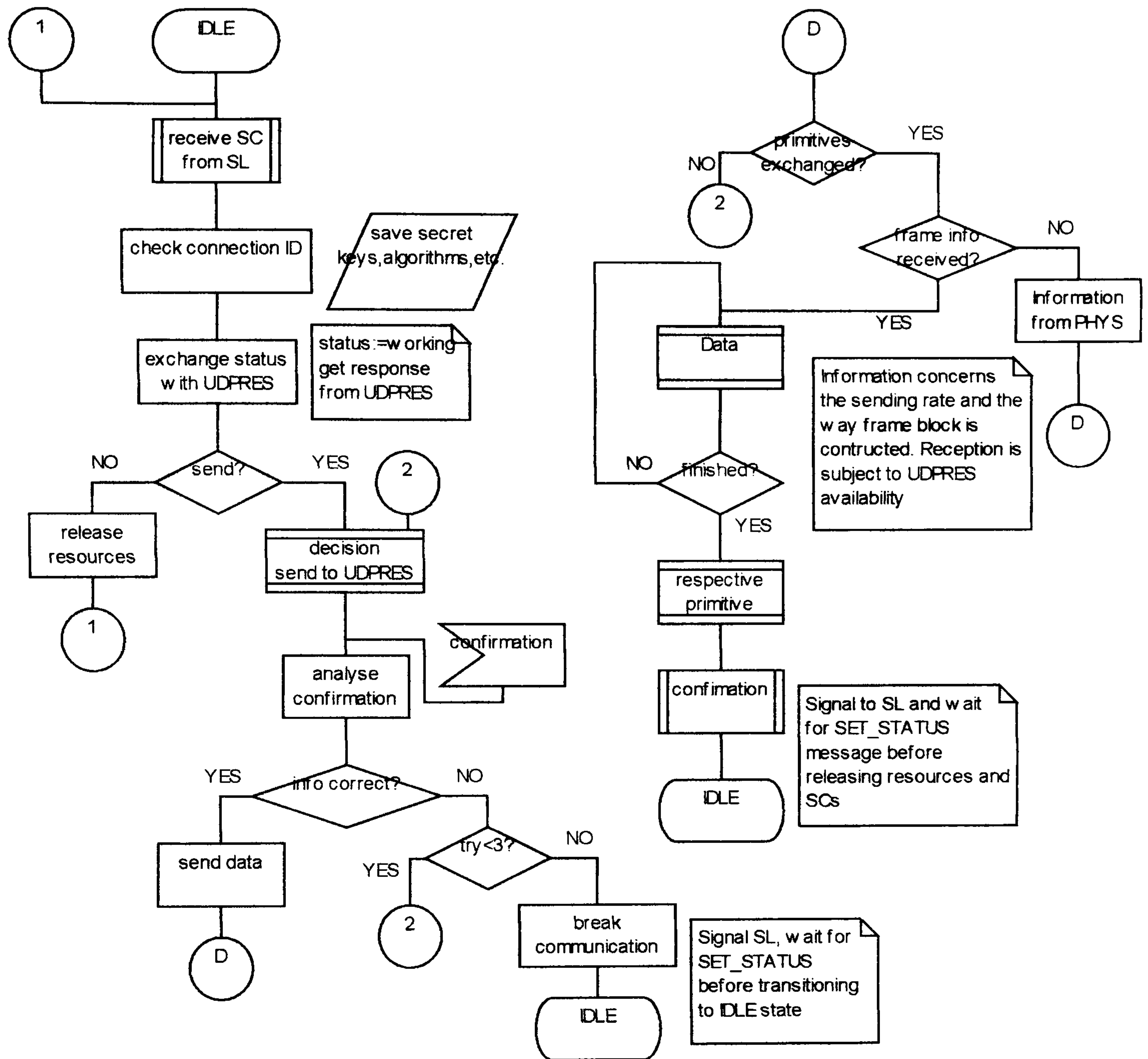


Figure A.2.2 – Node Receiving Instance

Process Type receiving

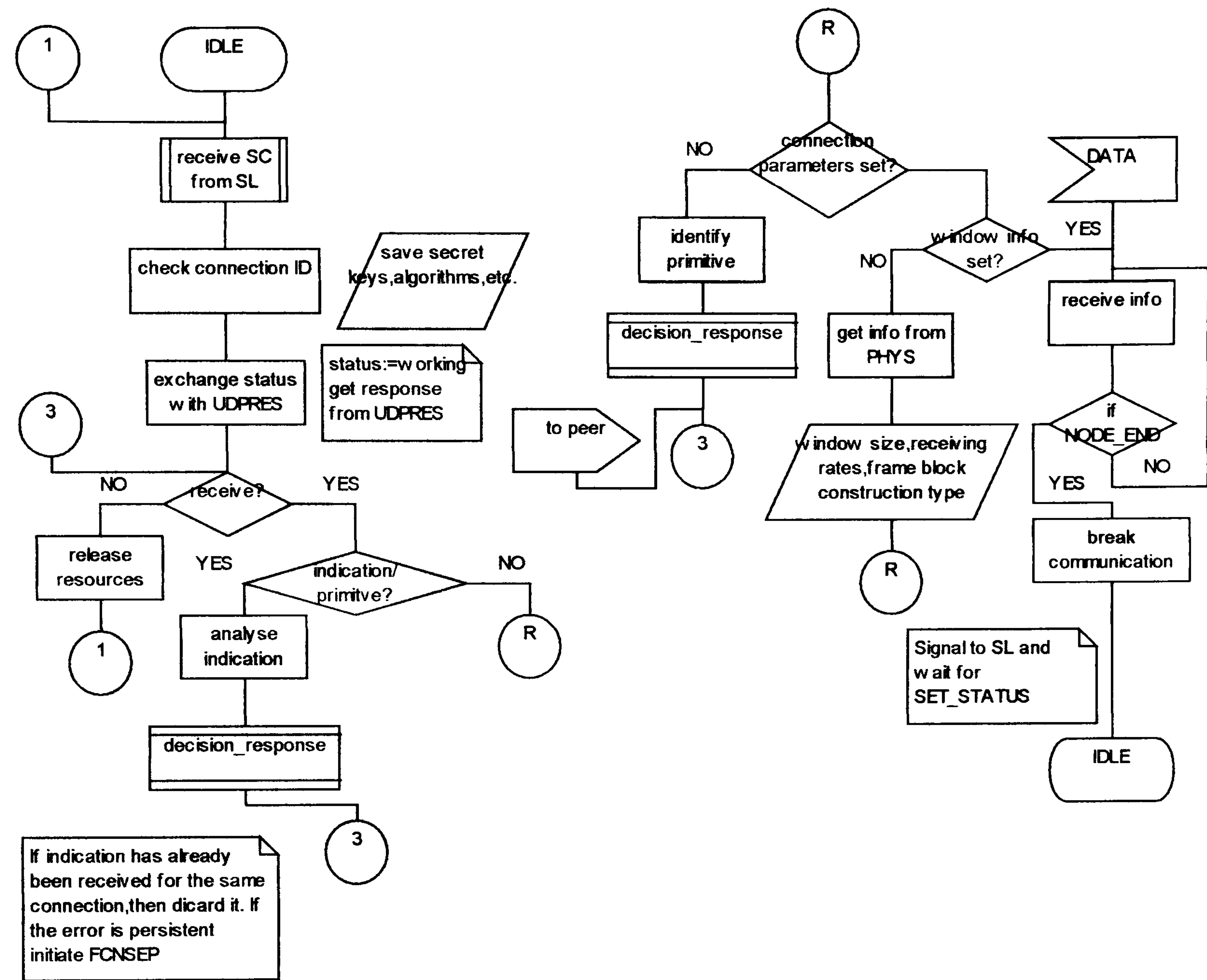


Figure A.3.1 – UDPRES Sending Instance

Process Type sending

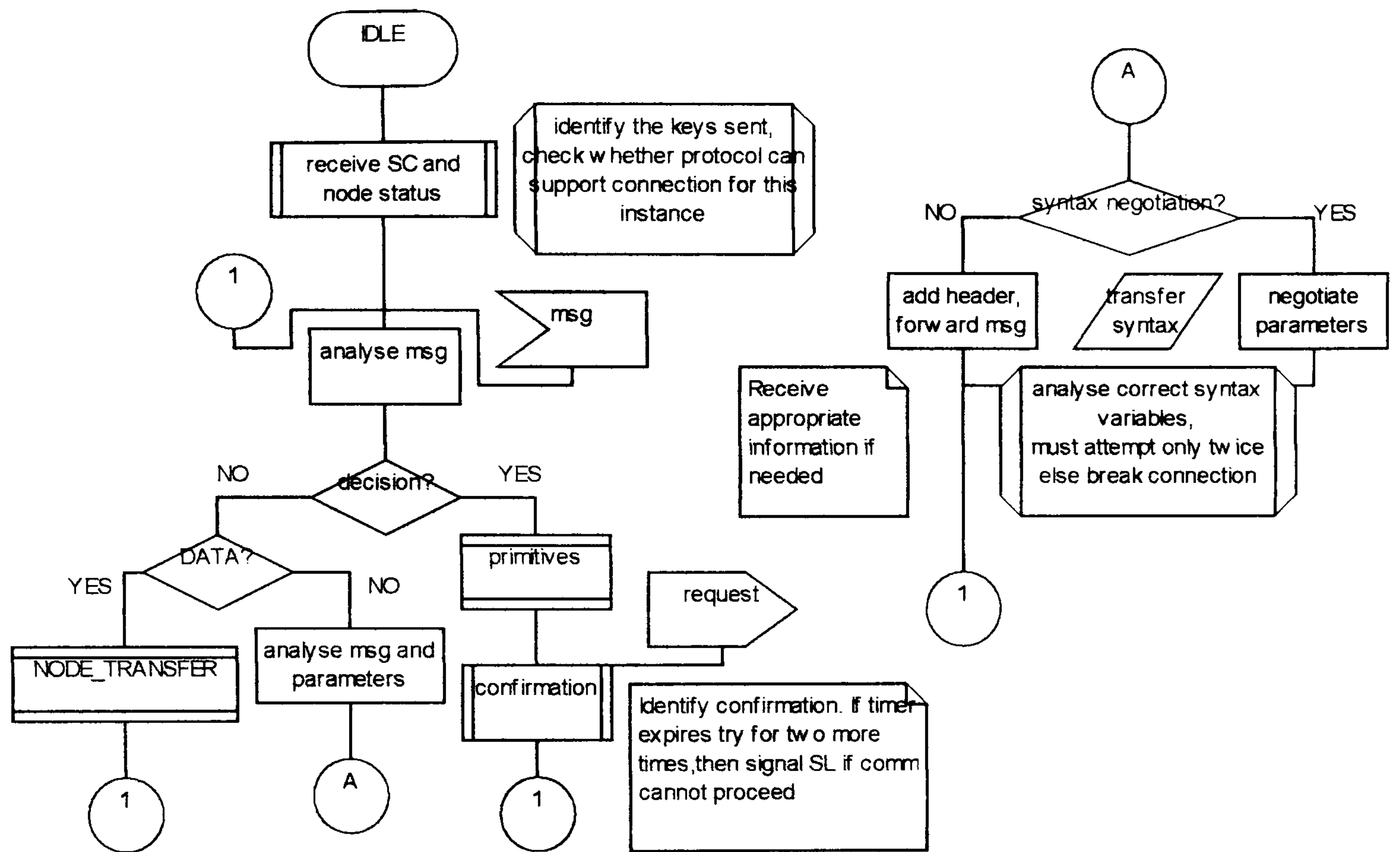


Figure A.3.2 – UDPRES Receiving Instance

Process Type receiving

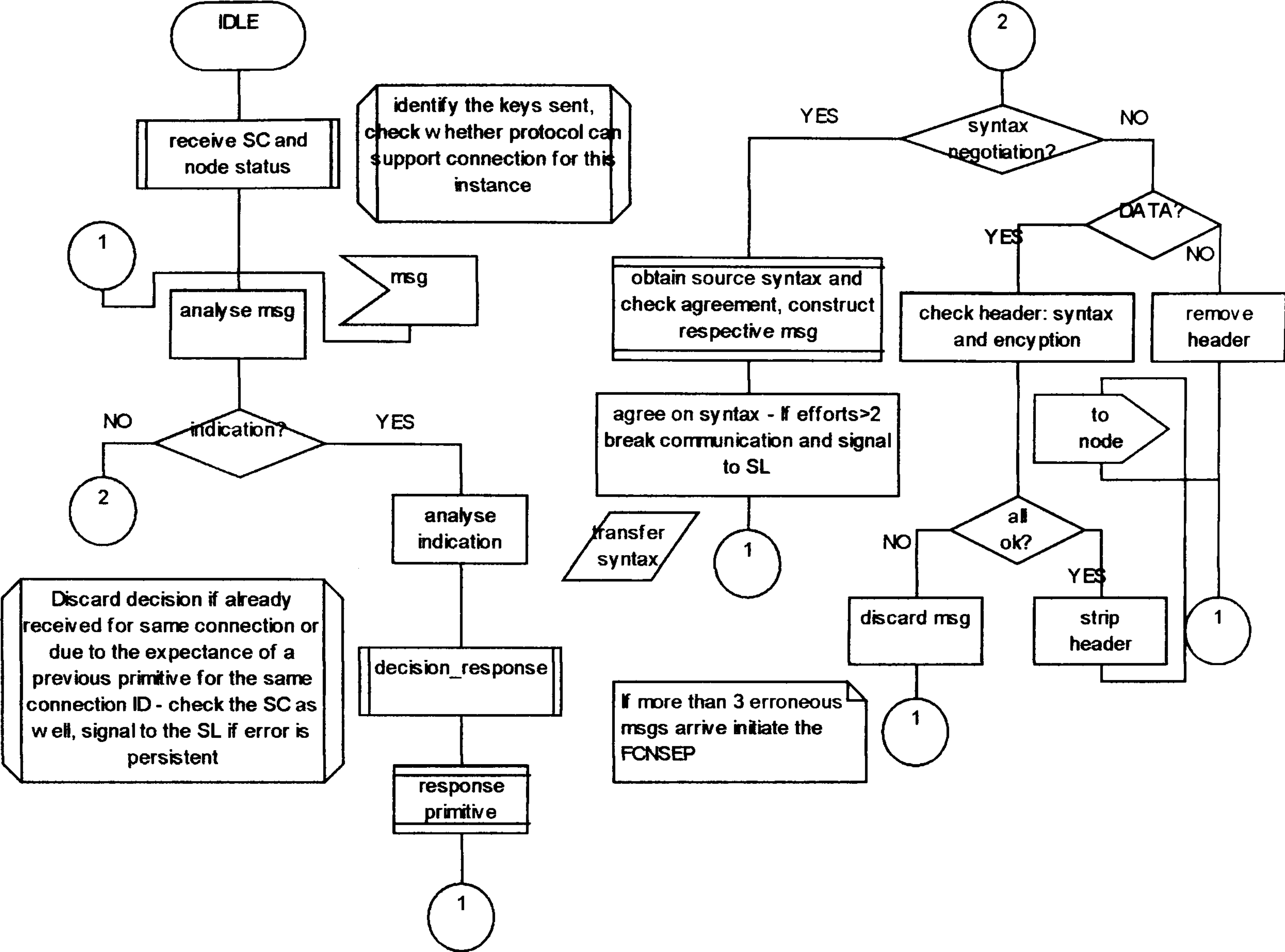


Figure A.4.1 – UDSES Sending Instance

Process Type sending

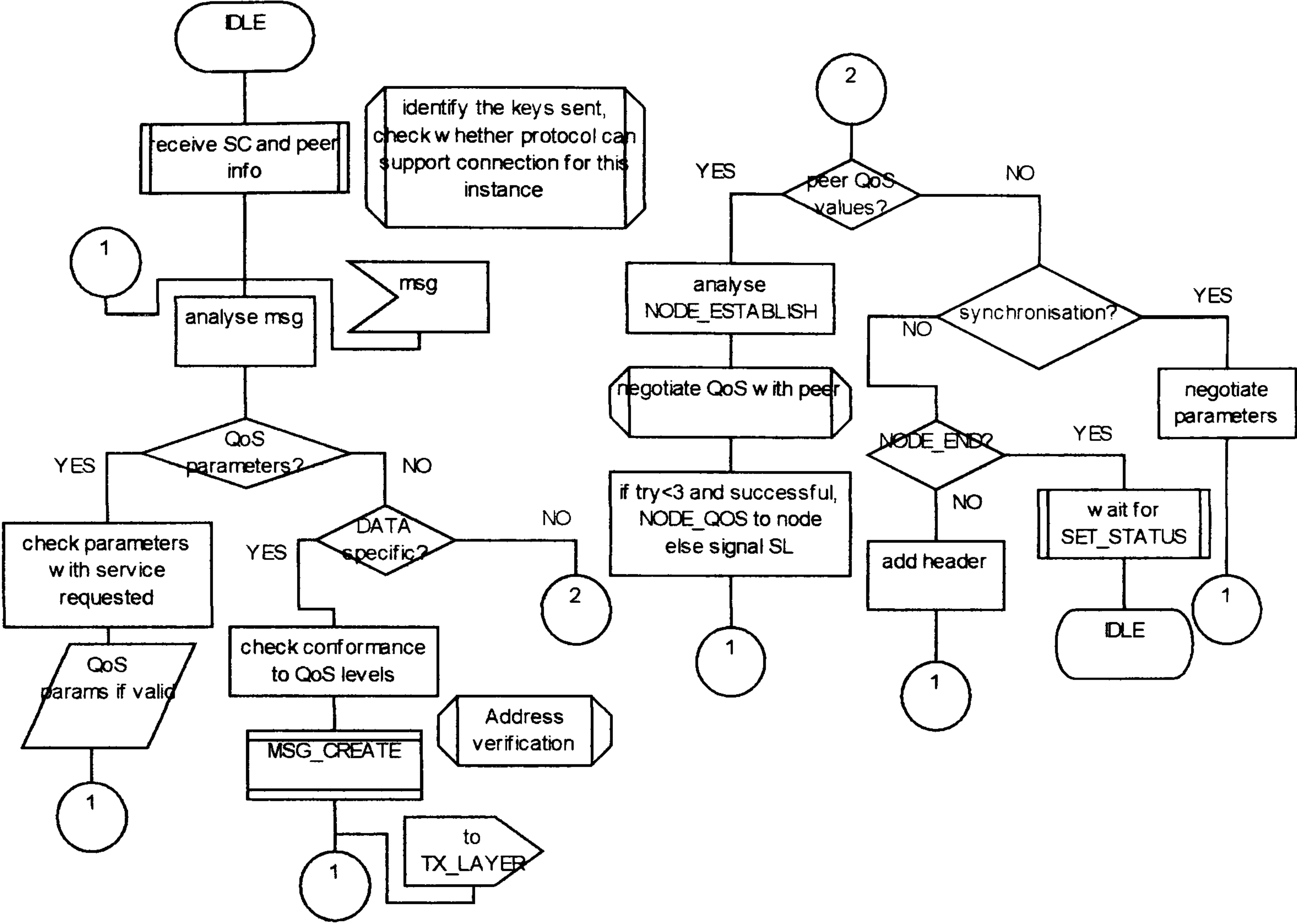


Figure A.4.2 – UDSES Receiving Instance

Process Type receiving

1(1 1):

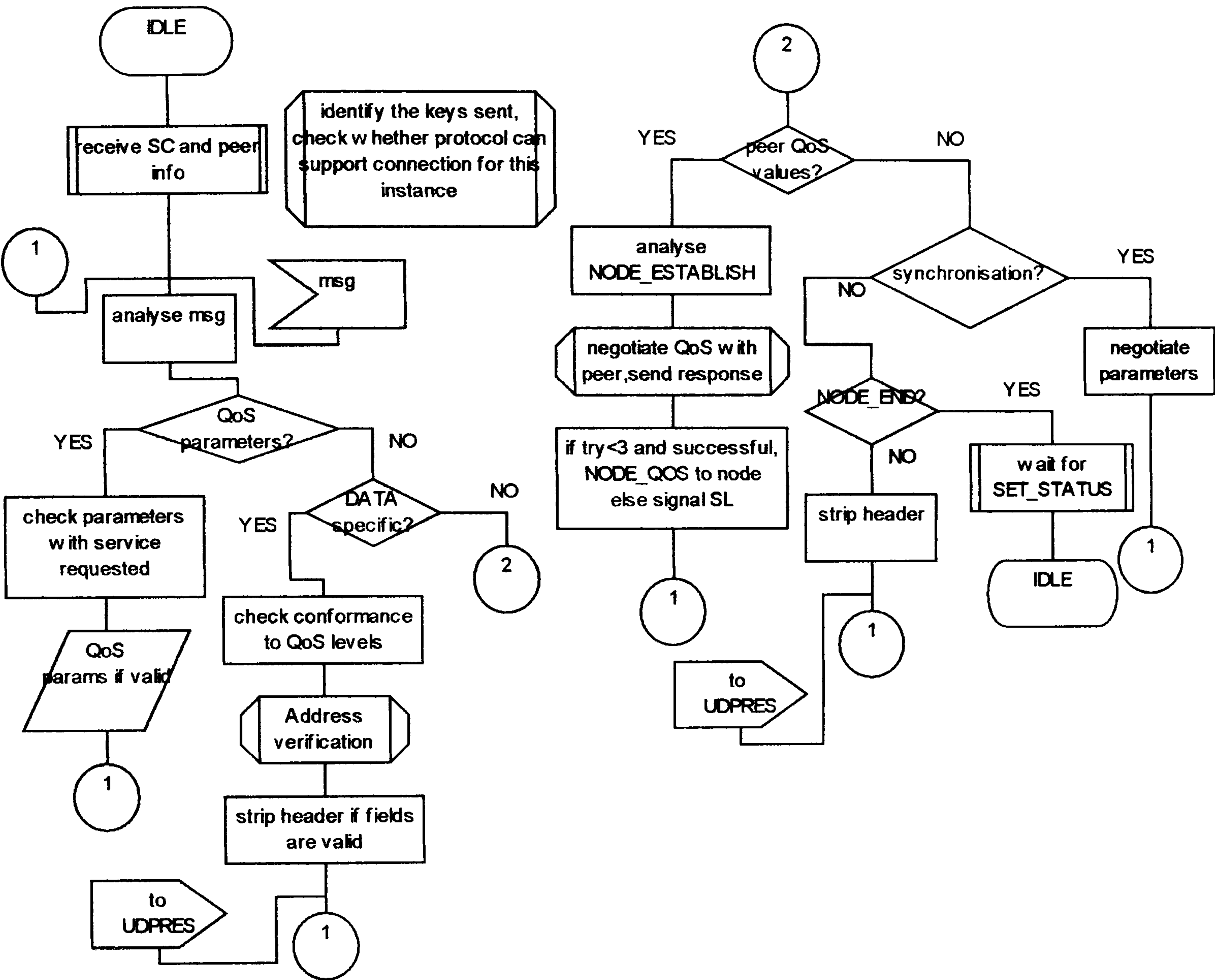


Figure A.5.1 – TX_LAYER Sending Instance

Process Type sending

1(1);1)

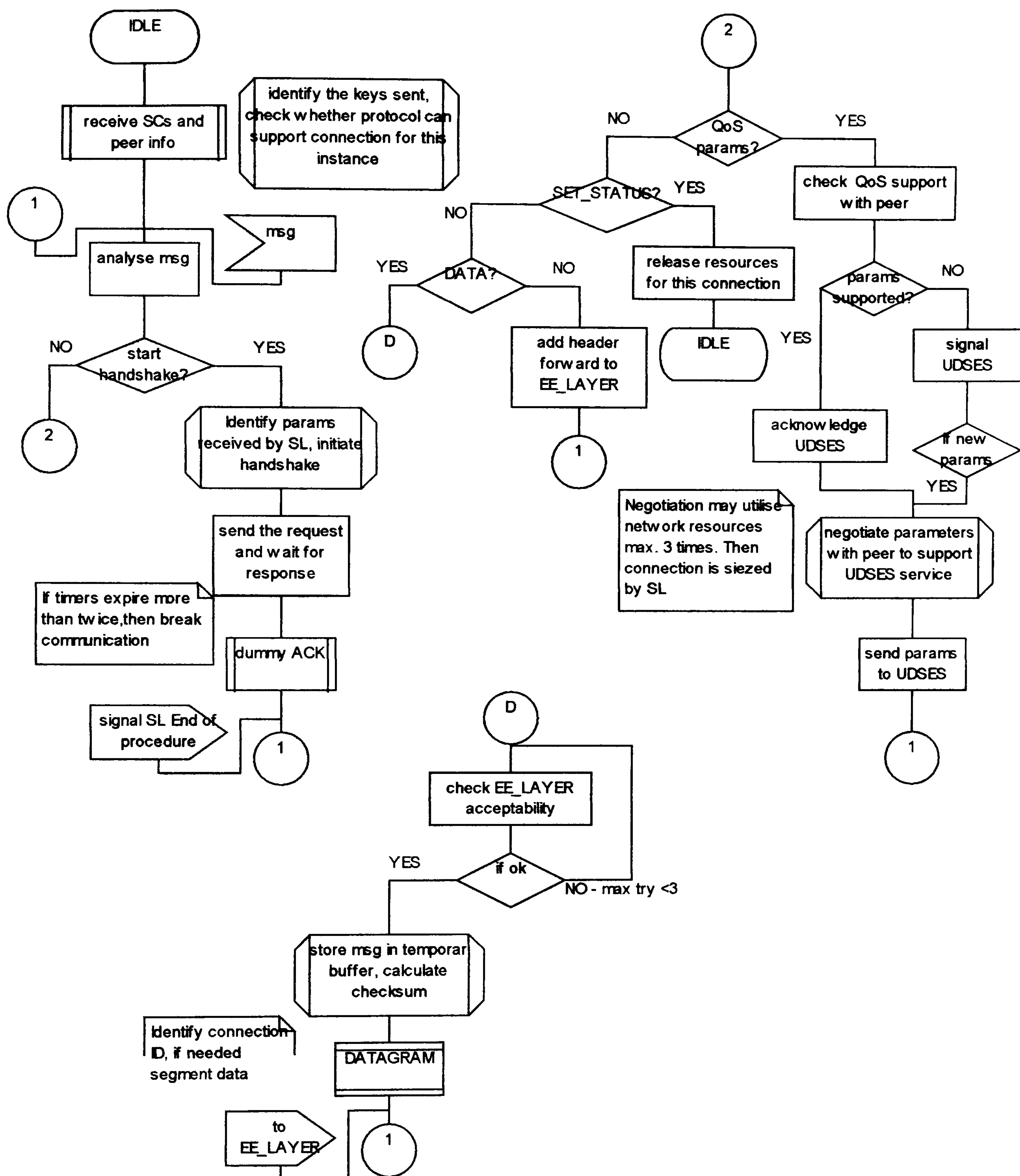


Figure A.5.2 – TX_LAYER Receiving Instance

Process Type receiving

1(1)(1)

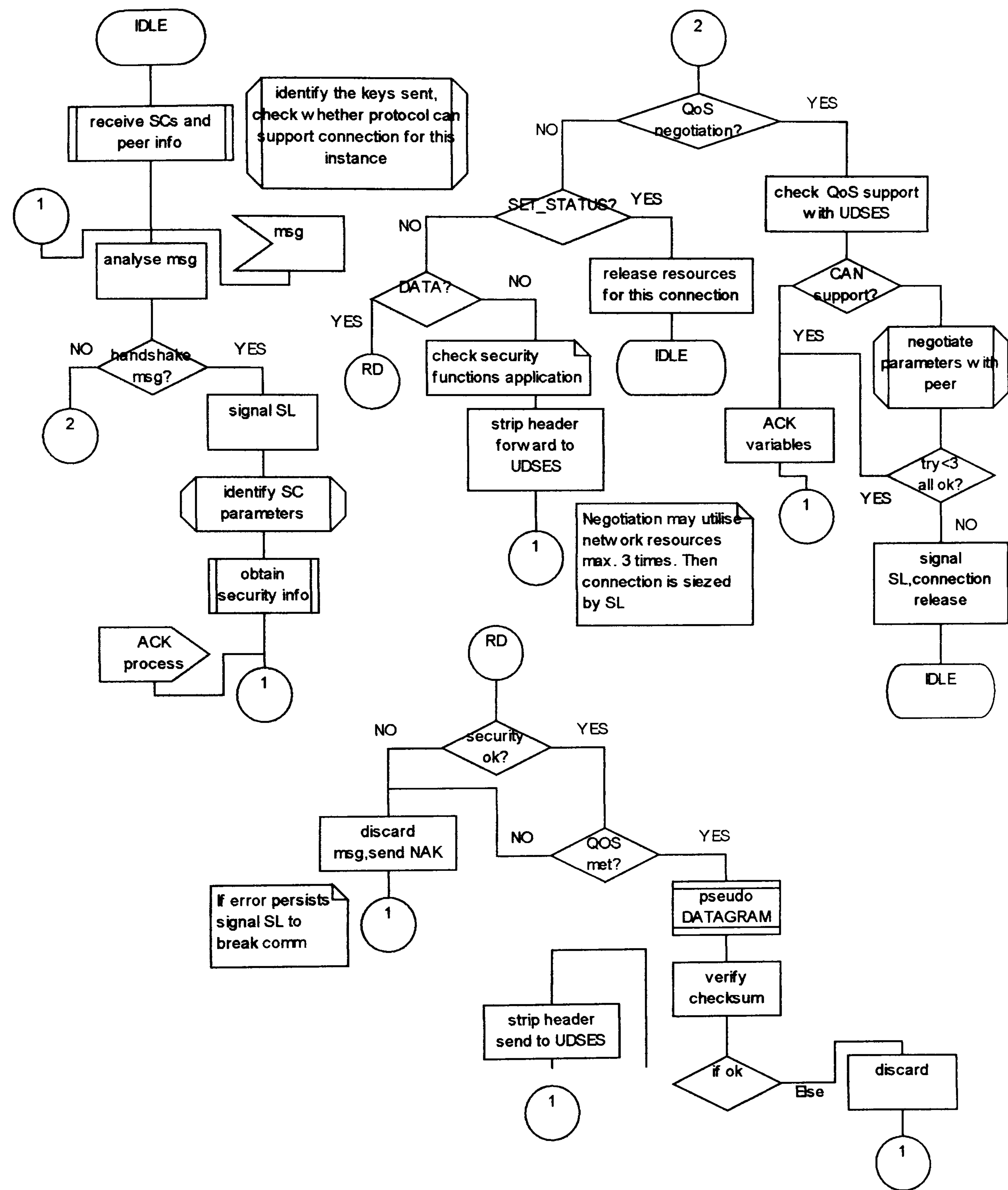
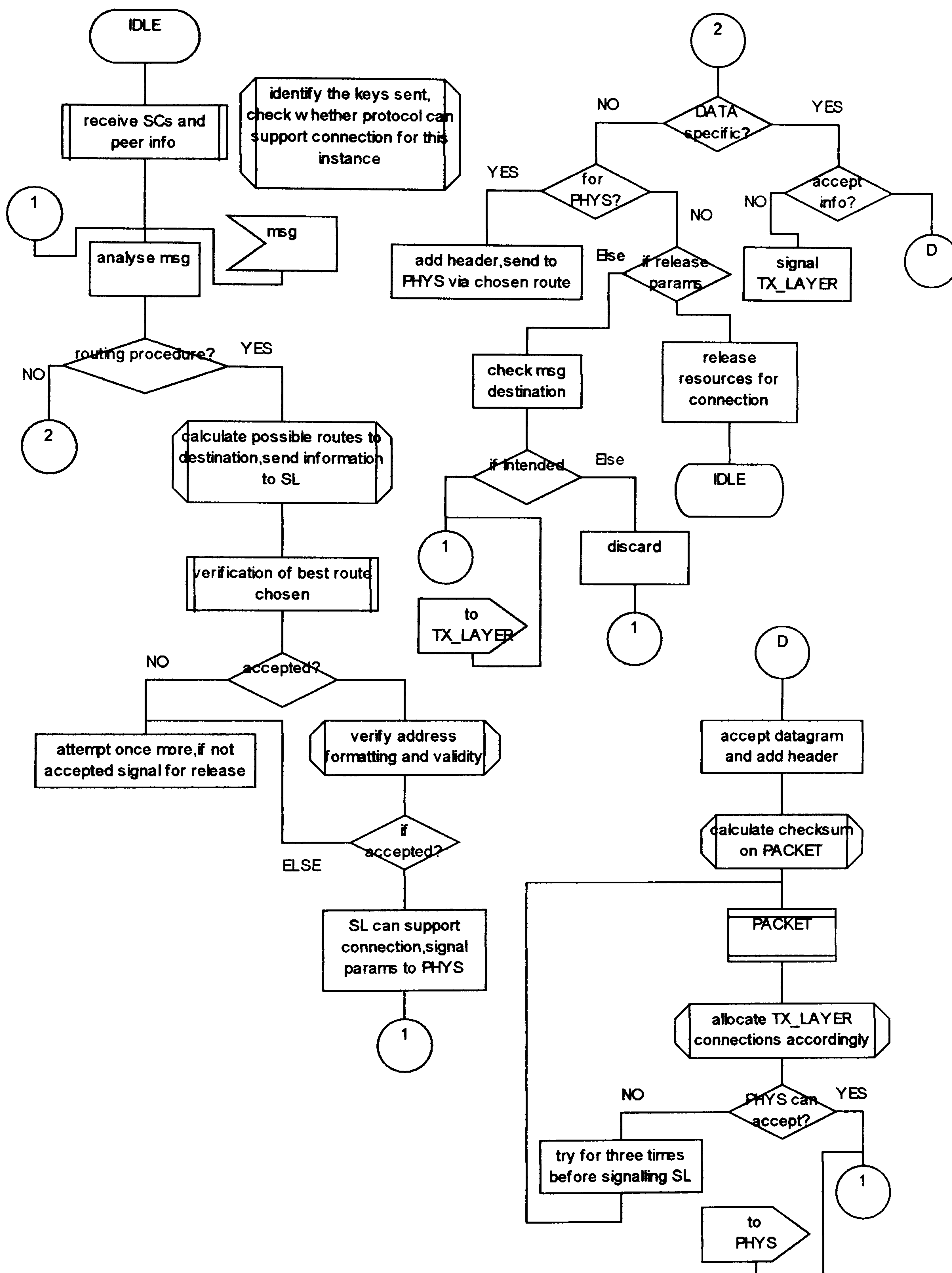


Figure A.6.1 – EE_LAYER Sending Instance

Process Type sending

1(1)1(1)



$1(1,1(1))$

```

graph TD
    IDLE1([IDLE]) --> RSC[receive SCs and peer info]
    RSC --> A1((1))
    A1 --> Analyse[analyse msg]
    Analyse --> RI1{route info?}
    RI1 -- NO --> A2((2))
    RI1 -- YES --> IRA[identify route and addresses sent by peer node]
    IRA --> IVA{if valid and accepted}
    IVA -- NO --> NAK[NAK, if try > 3 signal to SL for release]
    NAK --> A3((1))
    A3 --> TP1[/to peer node/]
    TP1 --> A1
    IVA -- YES --> AR[acknowledge reception]
    AR --> RI2[/route info/]
    RI2 --> SP[signal PHYS]
    SP --> A4((1))
    A4 --> IDLE2([IDLE])

    RD1((RD)) --> DATA{DATA?}
    DATA -- YES --> RD2((RD))
    DATA -- NO --> TL1{for TX_LAYER?}
    TL1 -- YES --> SH1[strip header send to TX_LAYER]
    SH1 --> A5((1))
    A5 --> IDLE3([IDLE])
    TL1 -- NO --> PH1{for PHYS?}
    PH1 -- YES --> AH[add header forward to PHYS]
    AH --> A6((1))
    A6 --> IDLE4([IDLE])
    PH1 -- NO --> SSTR[if SET_STATUS release resources, else discard]
    SSTR --> IDLE5([IDLE])

    RD3((RD)) --> FNT{for this node?}
    FNT -- YES --> CPH[construct pseudoheader, add DATAGRAM and calculate checksum]
    CPH --> CO{checksum ok?}
    CO -- YES --> SH2[strip header send to TX_LAYER]
    SH2 --> TP2[/to TX_LAYER/]
    TP2 --> A7((1))
    A7 --> IDLE6([IDLE])
    CO -- NO --> DM[discard msg]
    DM --> SLP[signal to SL if persistent]
    SLP --> A8((1))
    FNT -- NO --> DM
  
```


Figure A.7.1 – PHYS Sending Instance

Process Type sending

1(1)

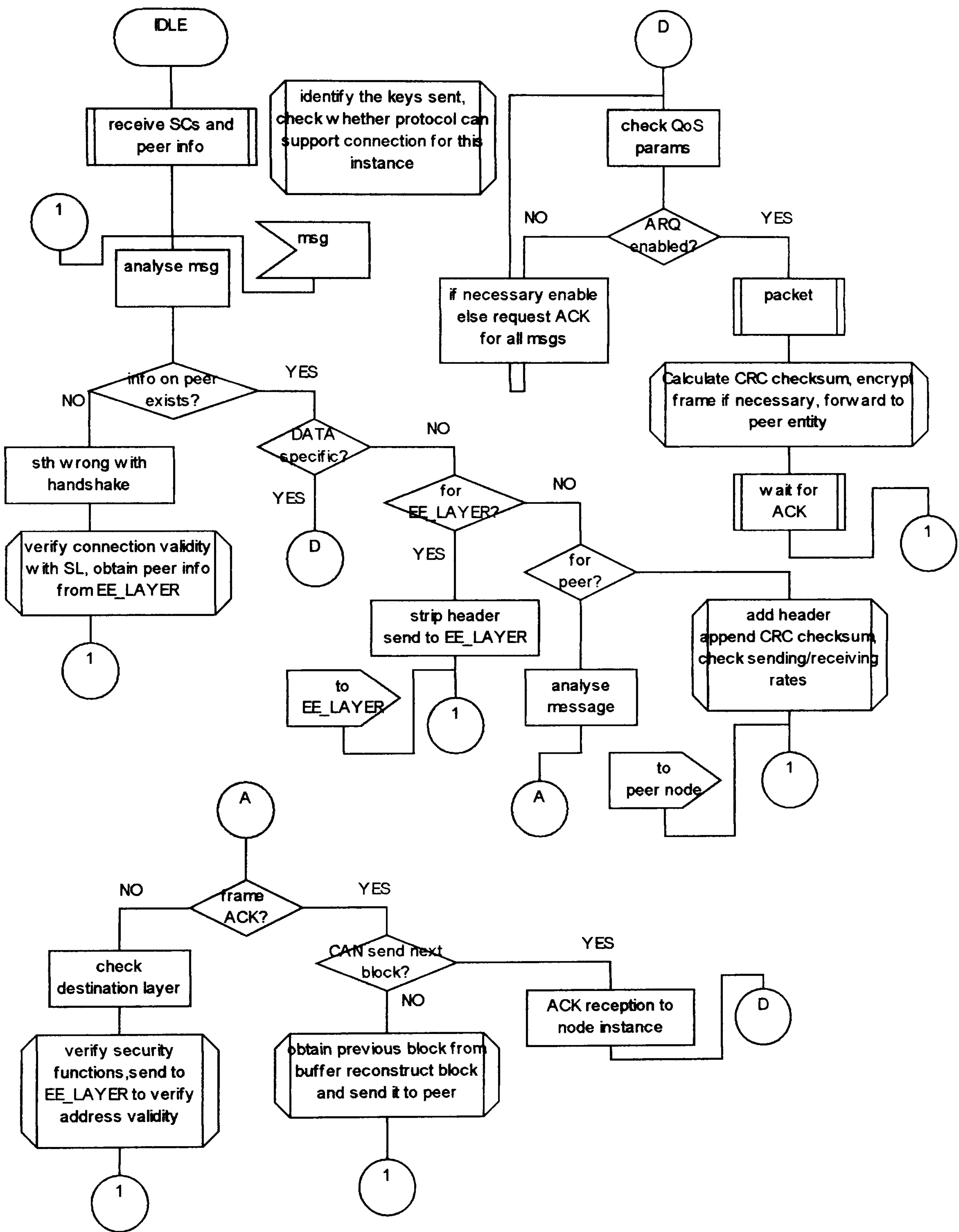


Figure A.7.2 – PHYS Receiving Instance

Process Type receiving

1(1)

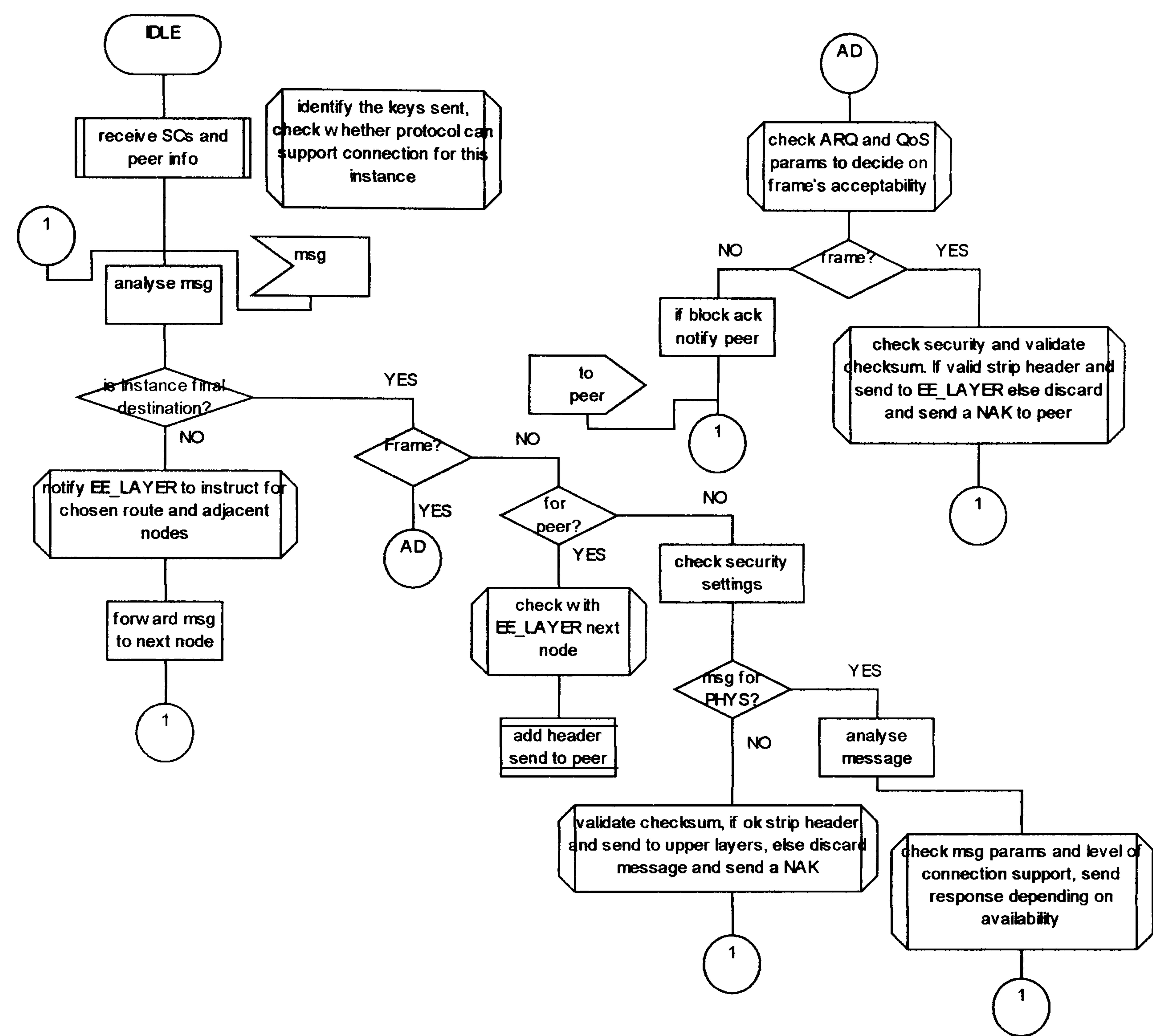


Figure A.8.1 – SL Sending Instance

Process Type sending

1(2)

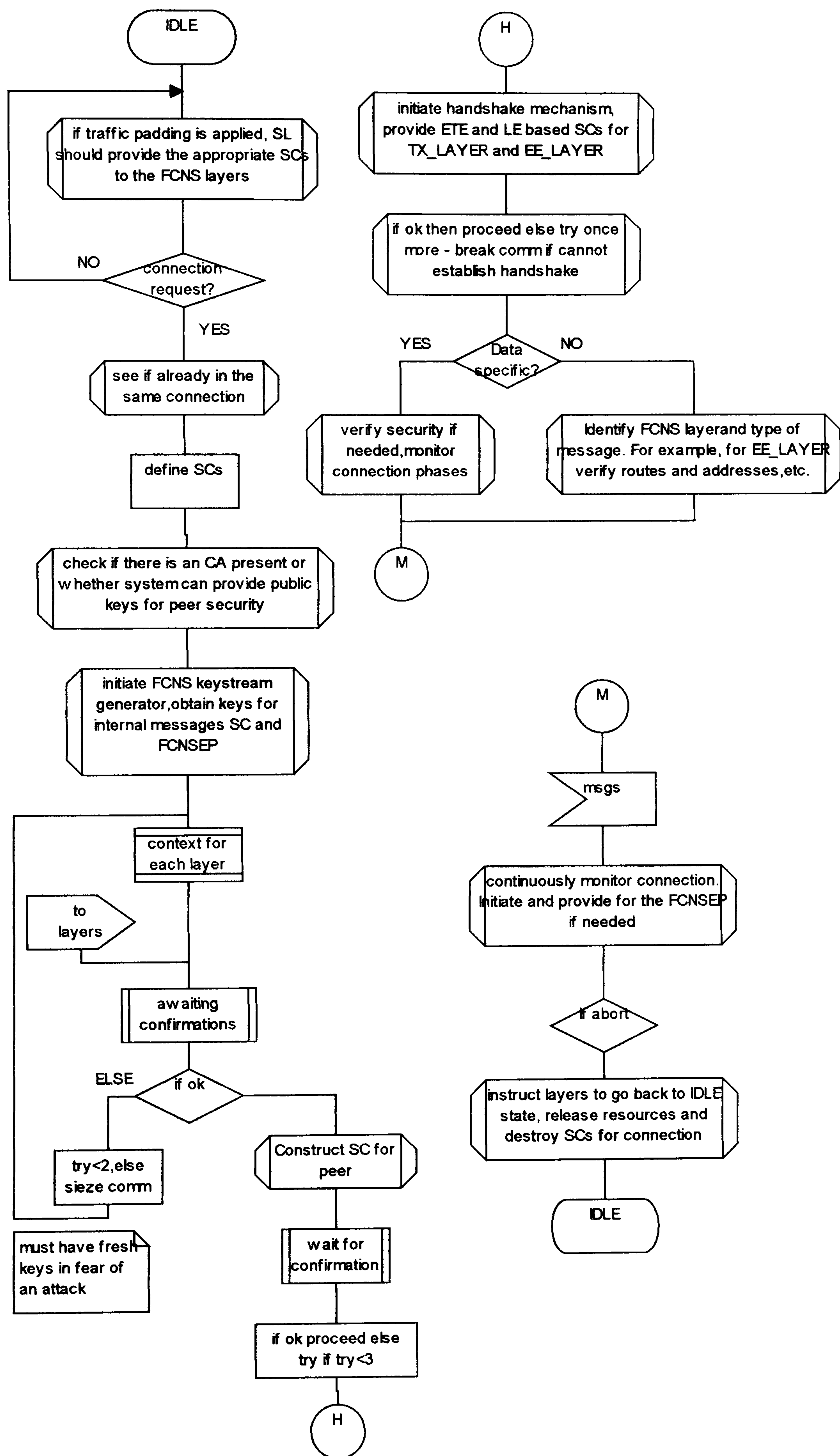


Figure A.8.2 – SL Receiving Instance

Process Type receiving

1(1)

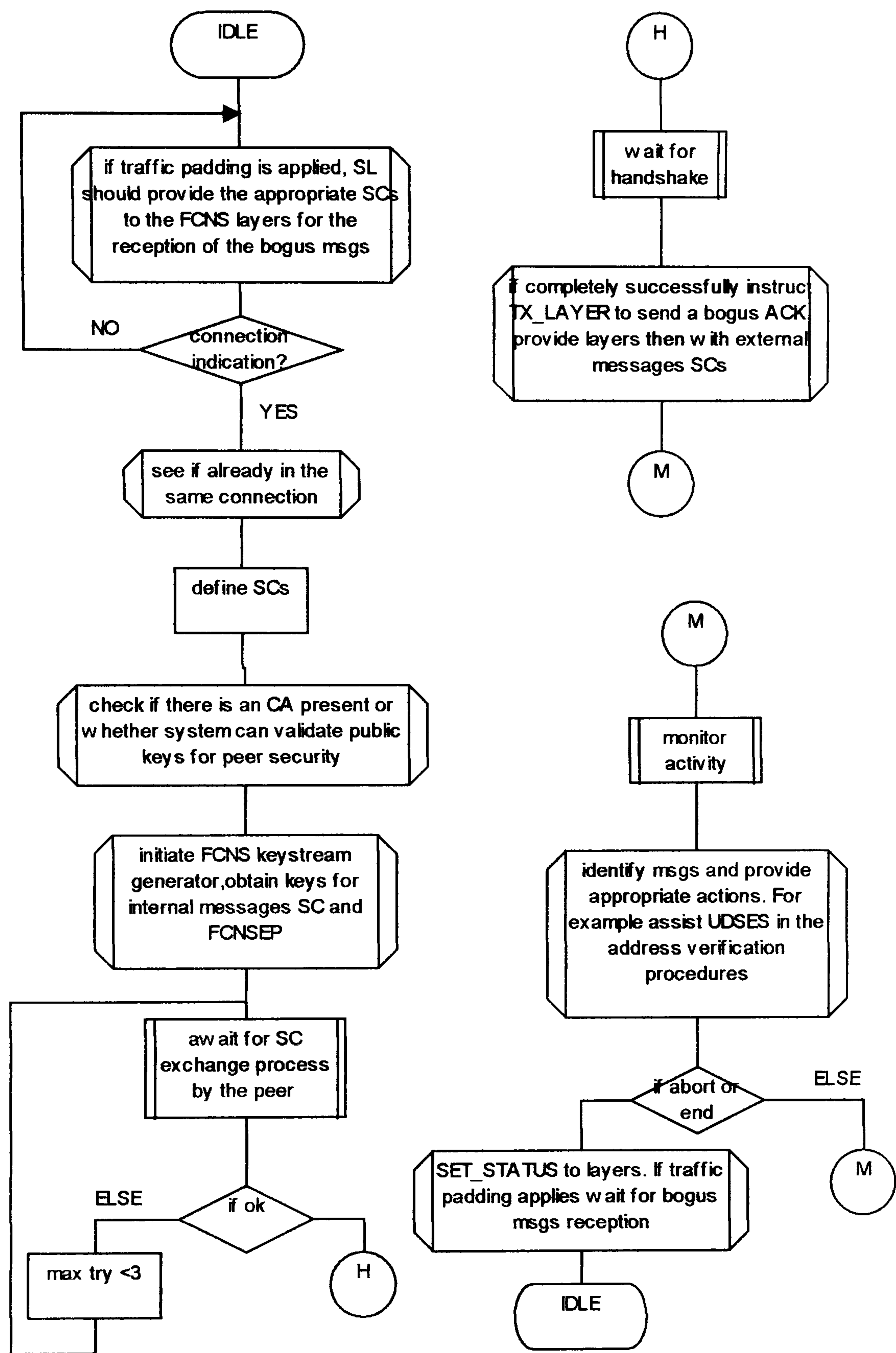


Figure A.9.1 – FCNSEP Sending Instance

Process Type sending

1(1)

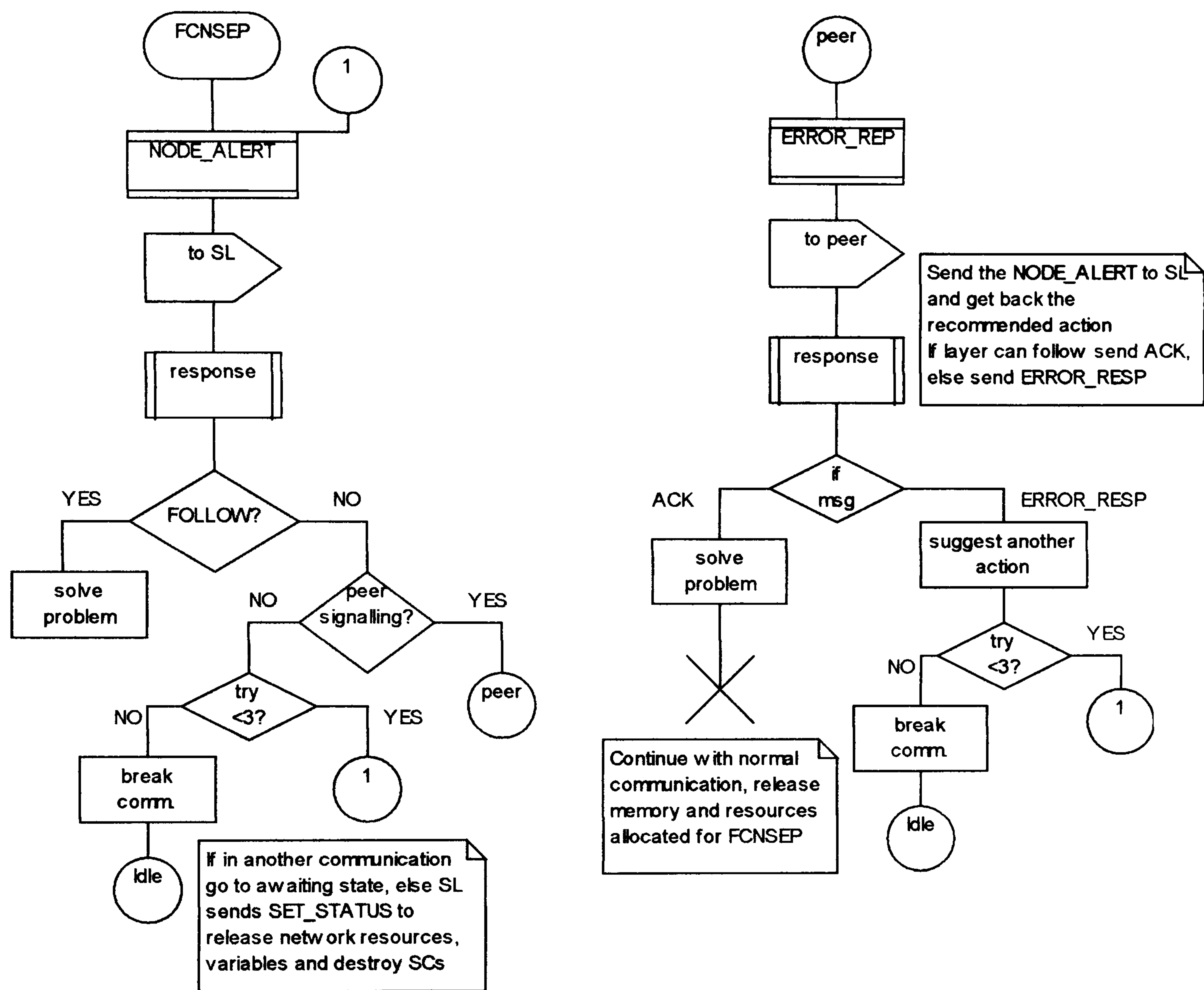
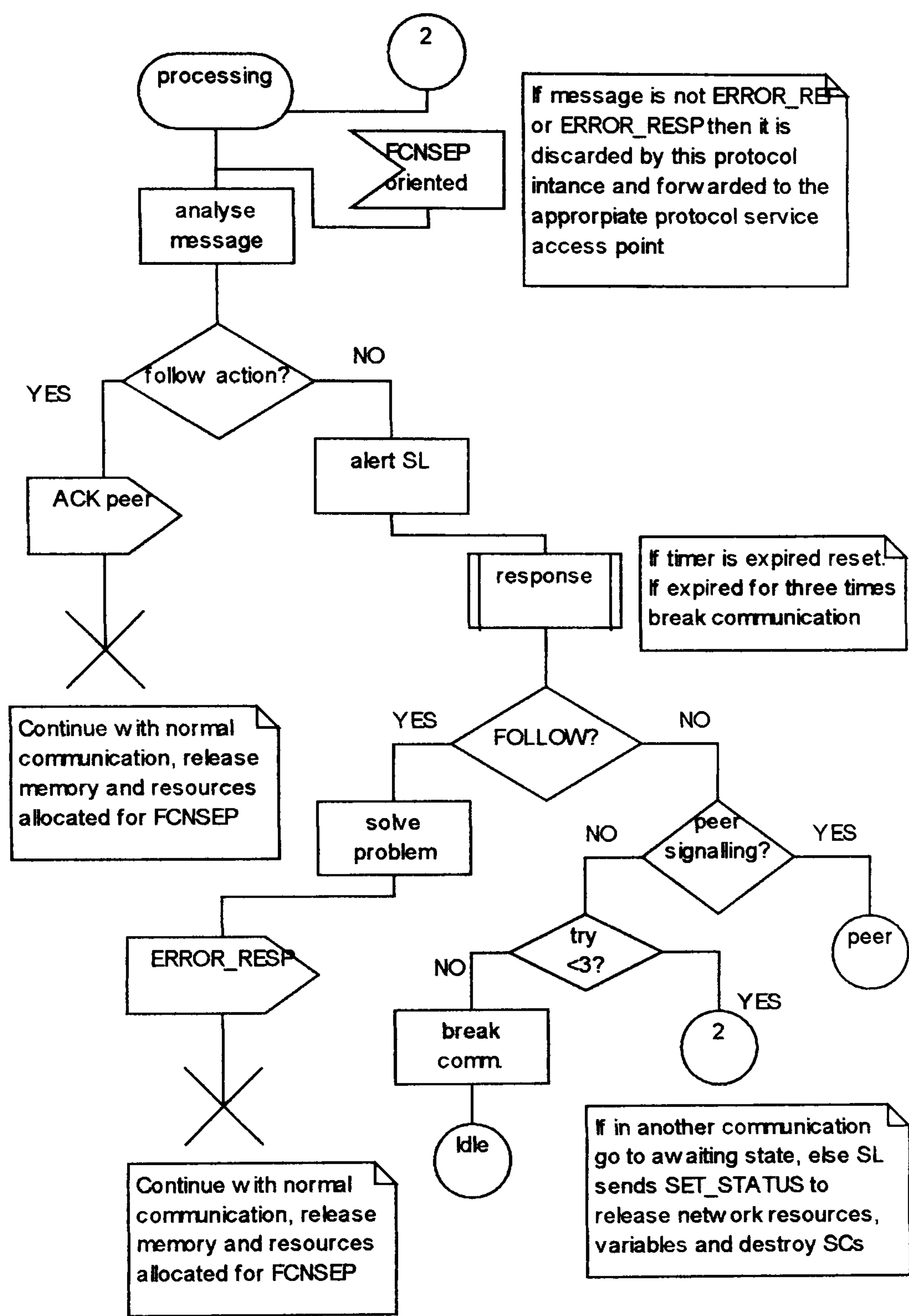


Figure A.9.2 – FCNSEP Receiving Instance

Process Type receiving

1(1)



Appendix B. FCNS Keystream Generator Statistical Test Results

Appendix B provides details of the results obtained by measuring the performance of the generator using the DIEHARD statistical tests battery. The generator has been made to produce more than 3,000,000 numbers, all placed on a binary file. The latter constitutes a prerequisite for the DIEHARD operation, which requires large files to enable the generation of a large sample of the output sequence.

Information is provided as to the measurements obtained for the following set of tests:

- Frequency (count-the-1s) tests – total of 2 tests by DIEHARD
- Poker (overlapping 5-permutation) test – total of 1 test
- Runs and long runs tests – total of 1 test
- Birthday test – total of 1 test
- Bitstream test – total of 1 test

For every test, the normal probability density function from the results obtained is depicted, together with the respective uniform cumulative density function (cdf). The normal distribution has been used since it models the *central limit theorem* [168]. This states that the sum of independent samples from any distribution with finite mean and variance converges to the normal distribution, as the sample size goes to infinity. The graphs are given to provide a measure identifying the response of the FCNS keystream generator to the tests completed, given the p values obtained. Figure B.1.1 illustrates the standard normal distribution reference graph whilst Figure B.1.2 the standard cdf of the (0,1) interval.

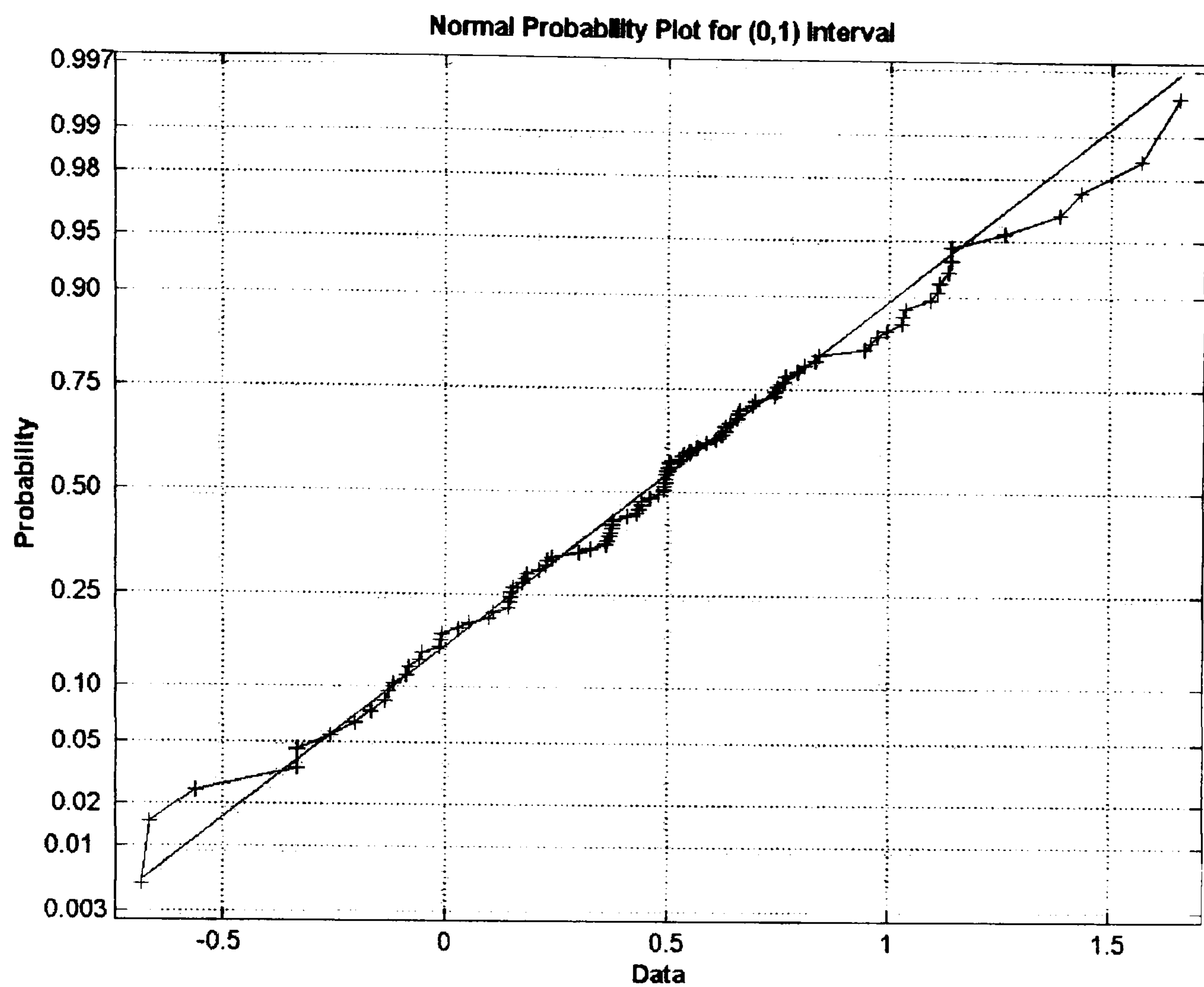


Figure B.1.1: Standard normal pdf

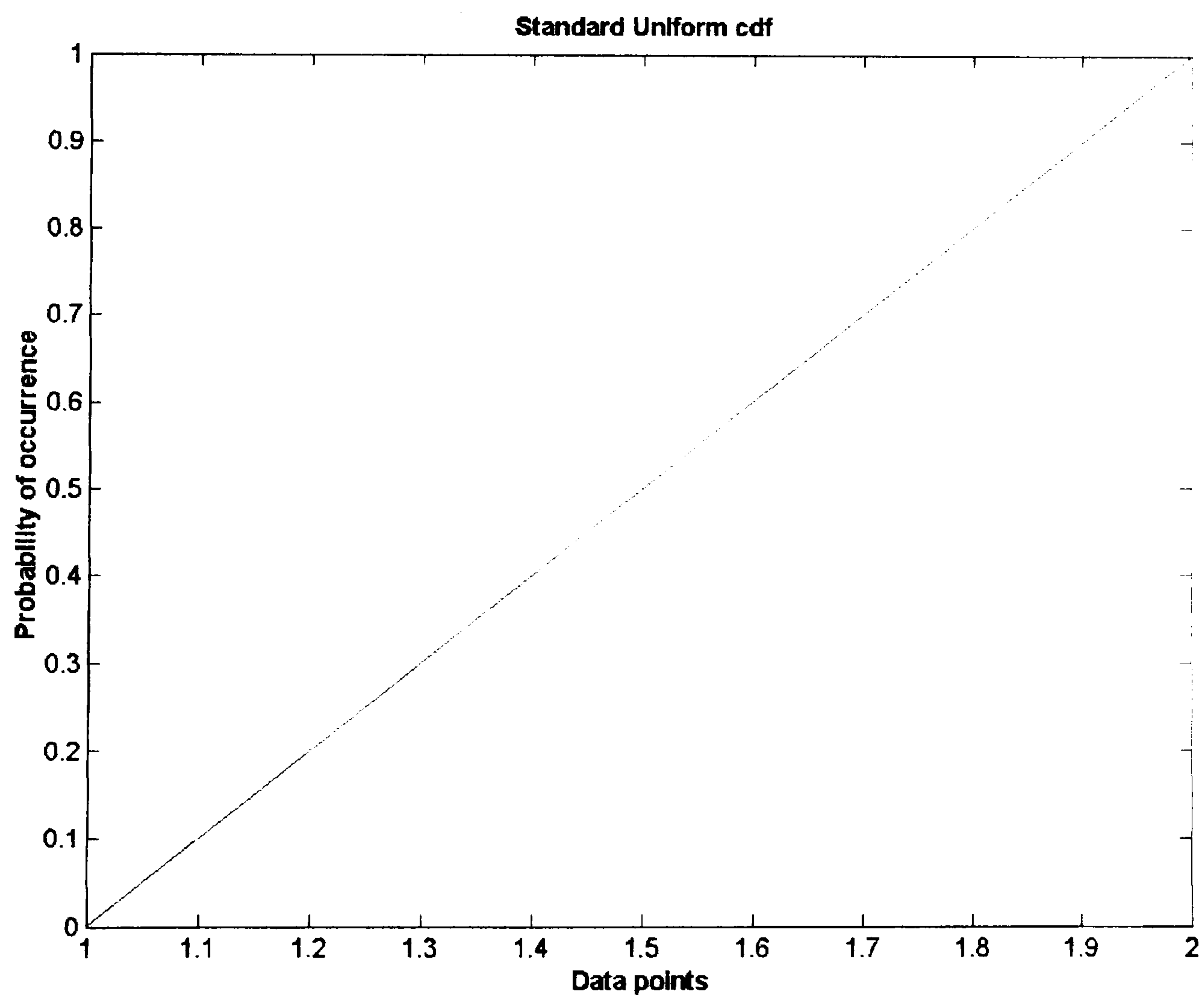


Figure B.1.2: Standard uniform cdf

B.1.1 MWCG of Equation 5-1 – 1st Test Set

Seed values: 229, 6709, 3929

Carry values: 983, 1181, 53

Multipliers: $a_1 = 2111111111$, $a_2 = 2131995753$, $a_3 = 1517746329$, $b_1 = 1447497129$, $b_2 = 24456079$, $b_3 = 94901263$

Key obtained: b9bc16f8 6531693d 6b9a718c efaf19ed 17194a48

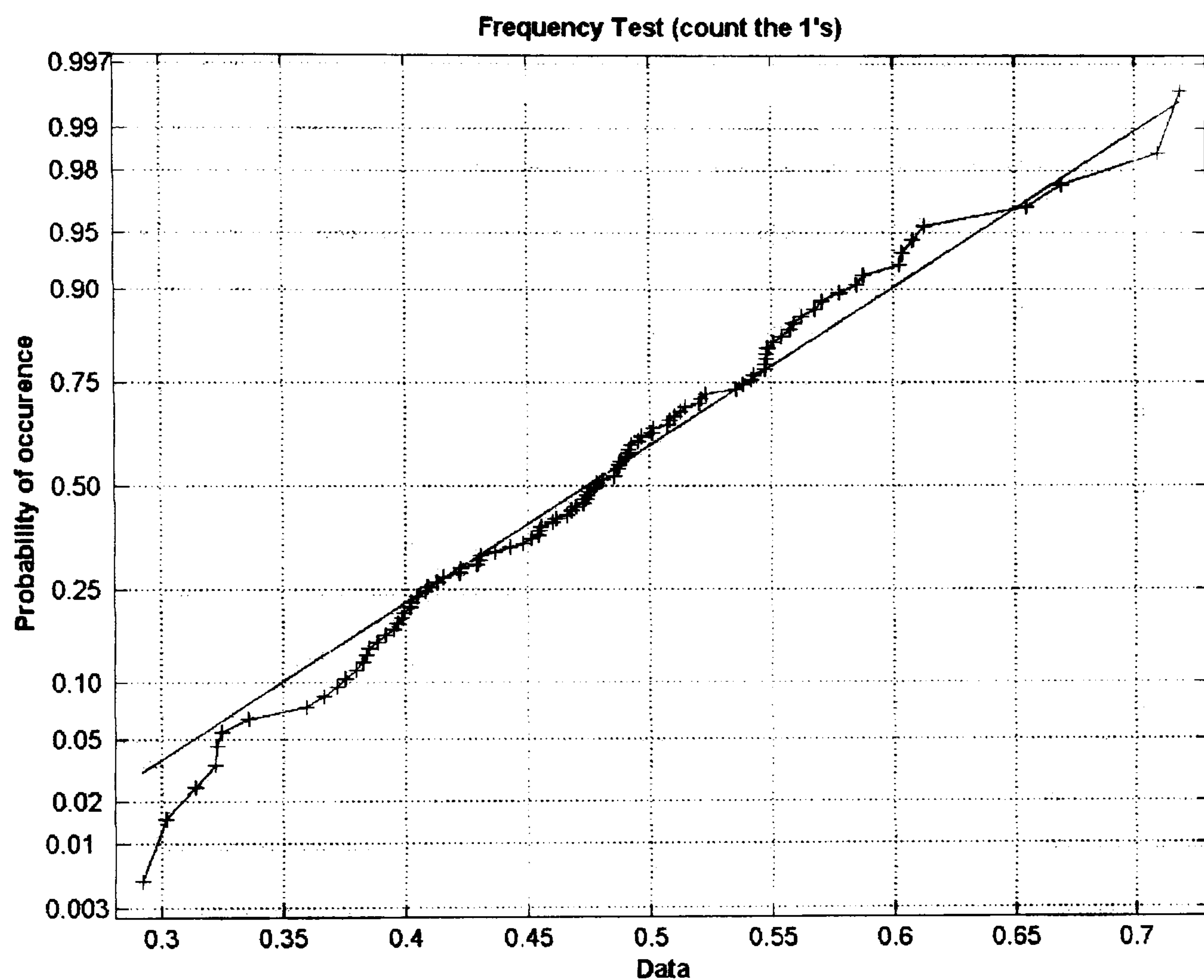


Figure B.2.1: Frequency test normal pdf – first MWCG, first test

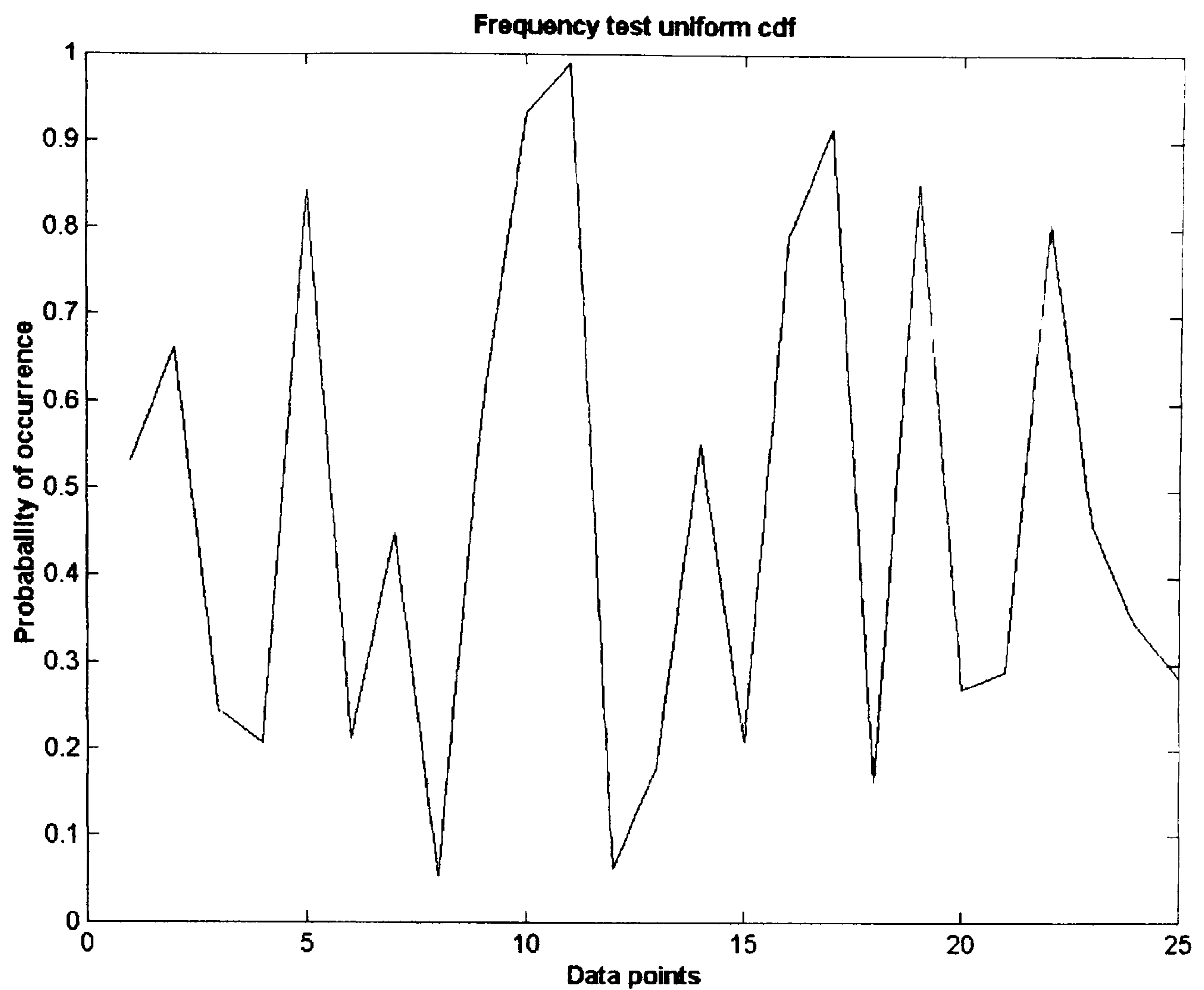


Figure B.2.2: Frequency test uniform cdf – first MWCG, first test

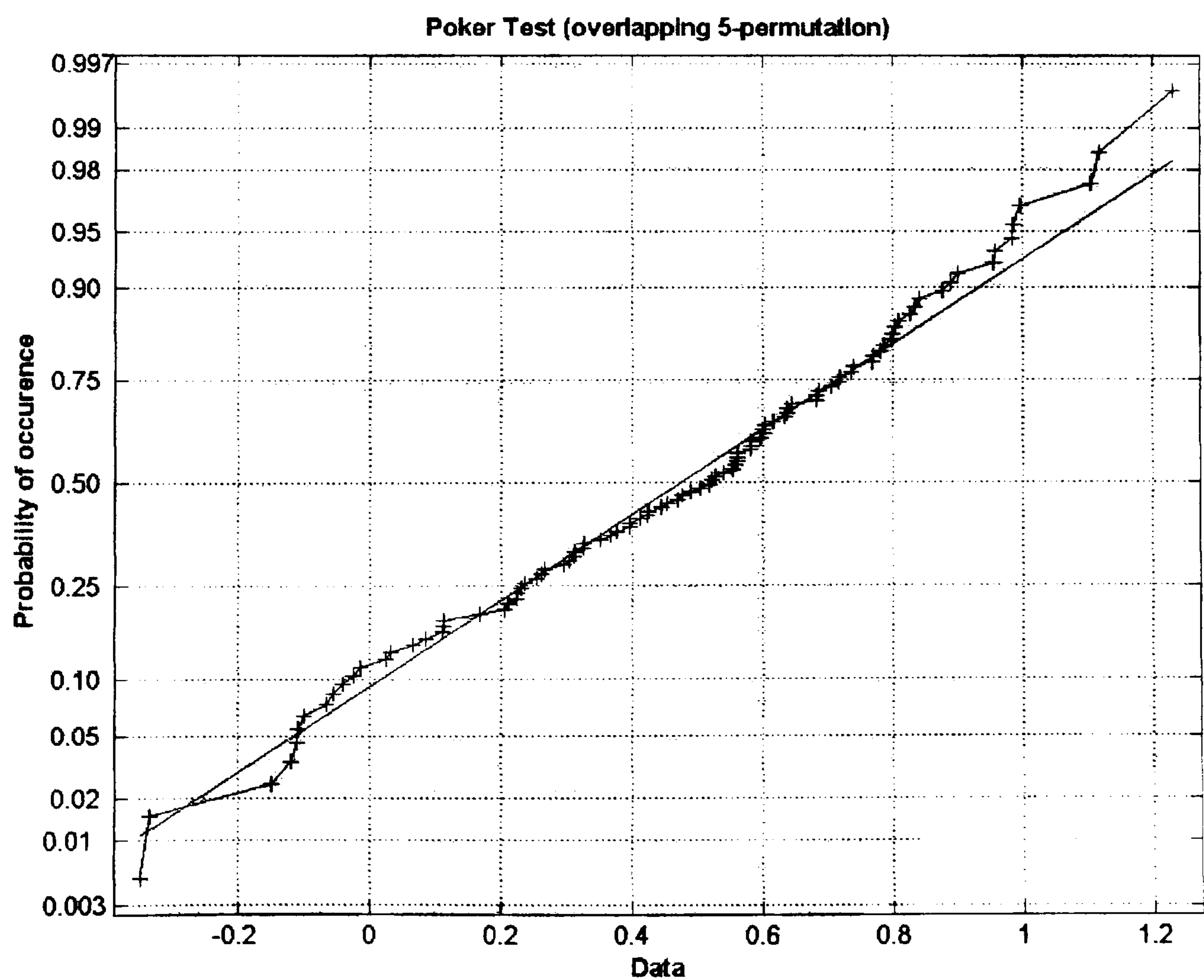


Figure B.3.1: Poker test normal pdf – first MWCG, first test

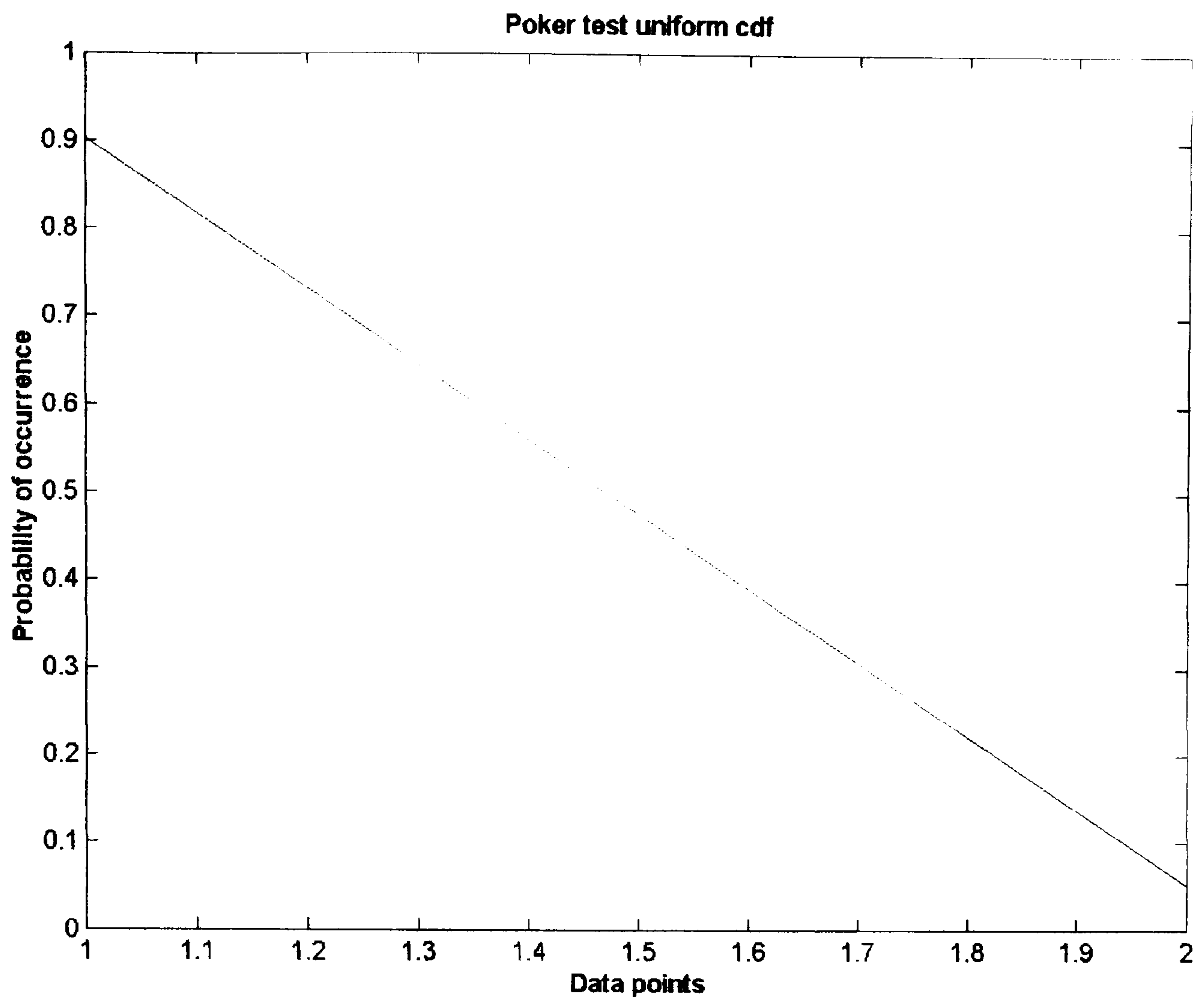


Figure B.3.2: Poker test uniform cdf – first MWCG, first test

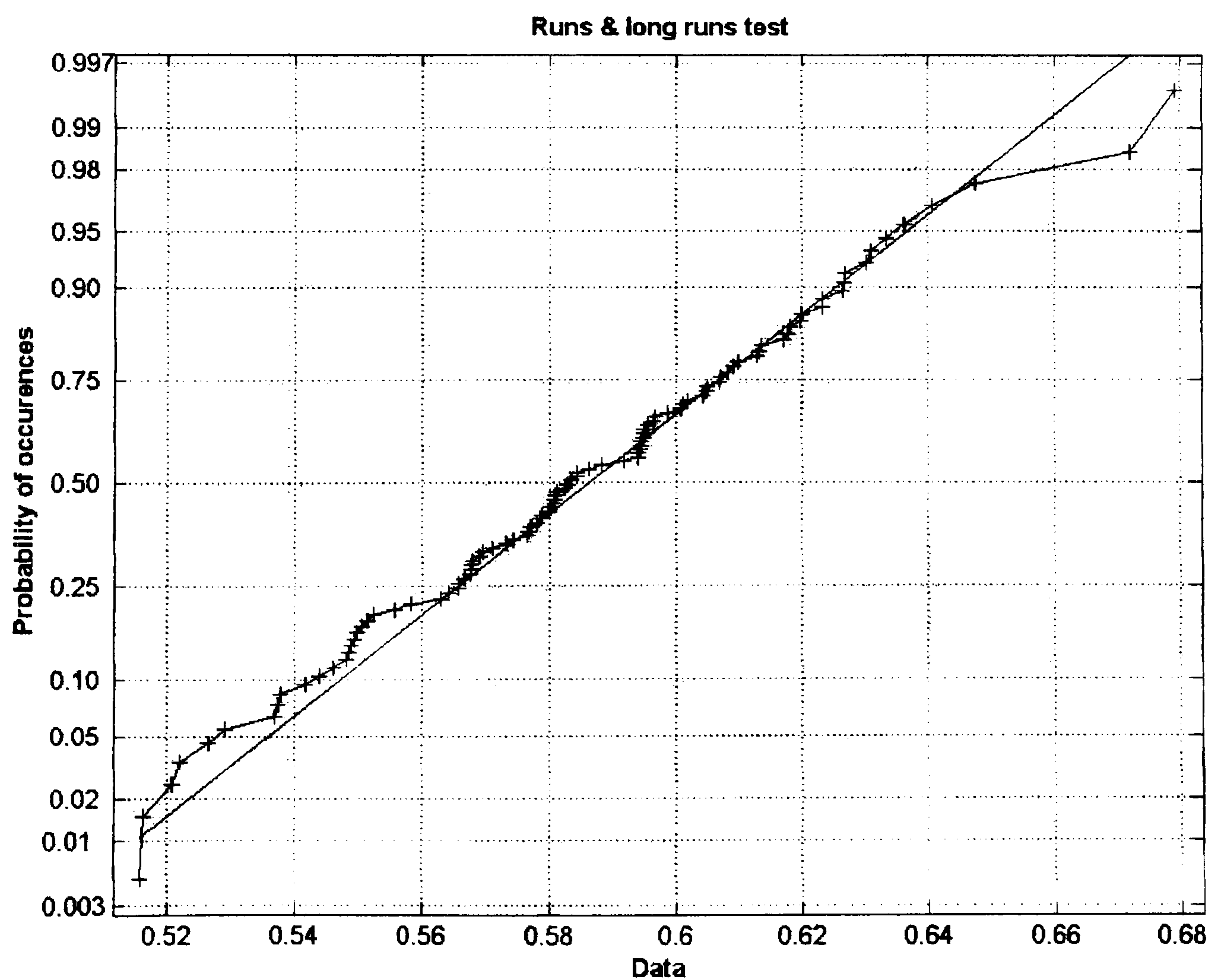


Figure B.4.1: Runs and long runs tests normal pdf – first MWCG, first test

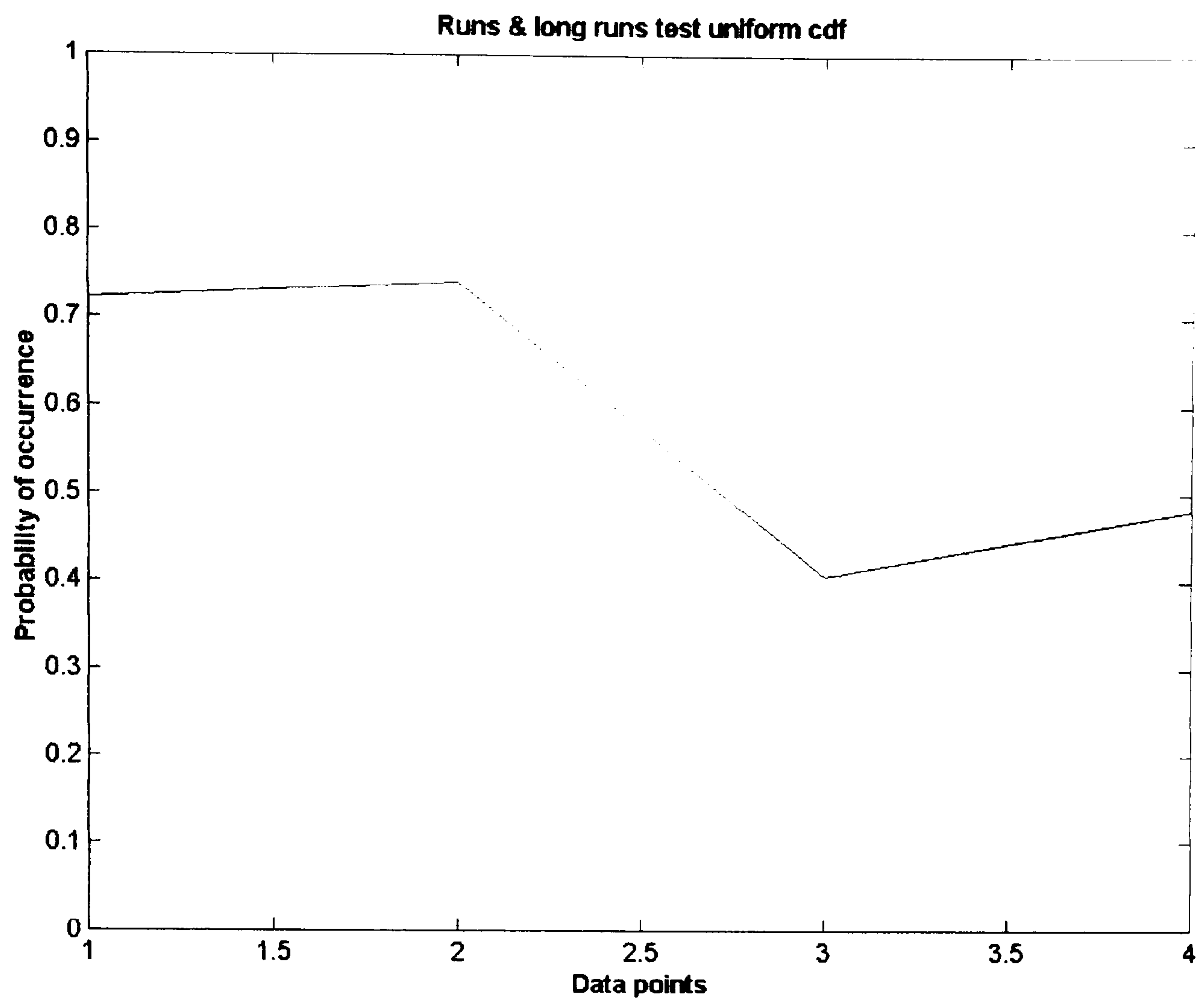


Figure B.4.2: Runs and long runs tests uniform cdf – first MWCG, first test

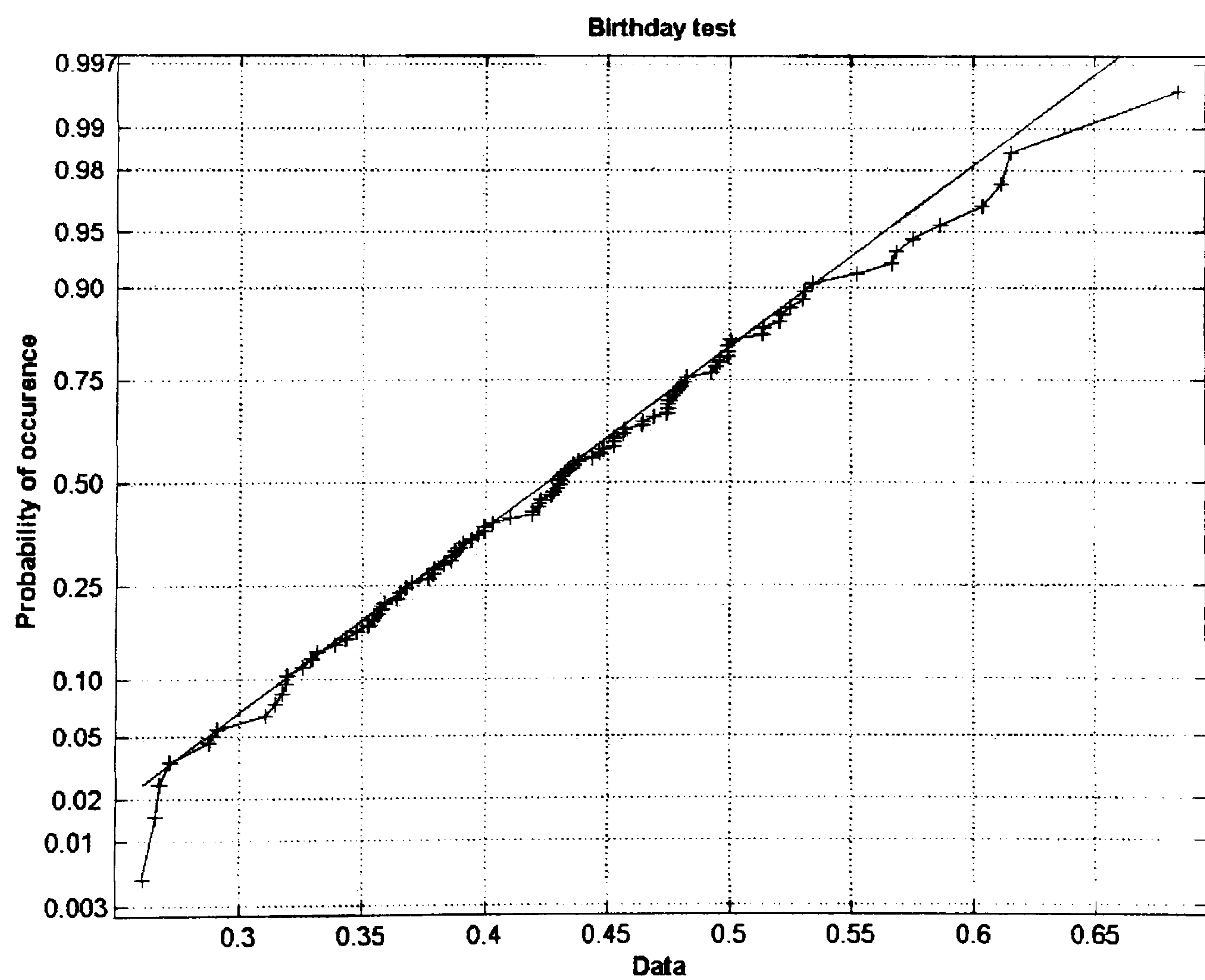


Figure B.5.1: Birthday test normal pdf – first MWCG, first test

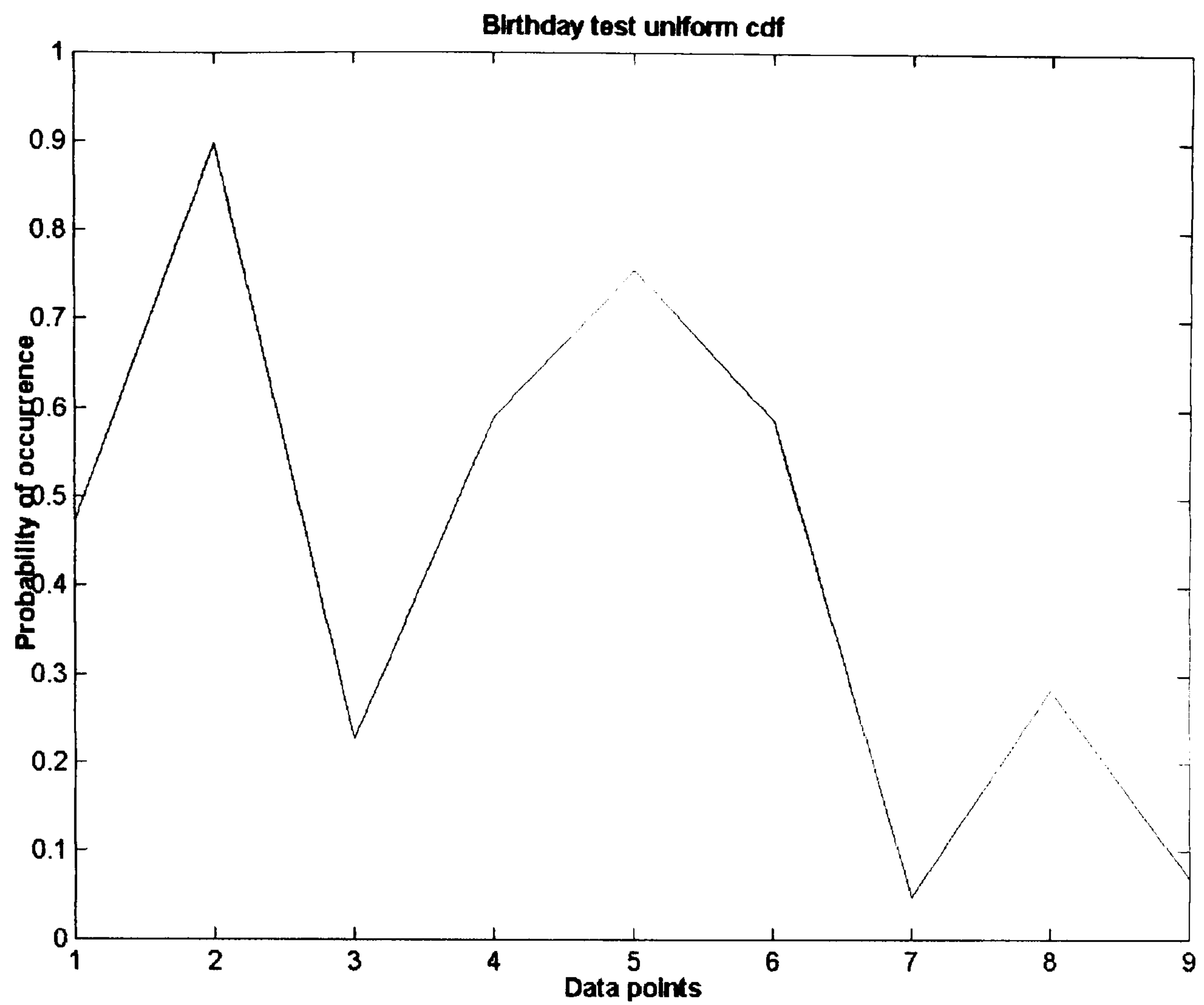


Figure B.5.2: Birthday test uniform cdf – first MWCG, first test

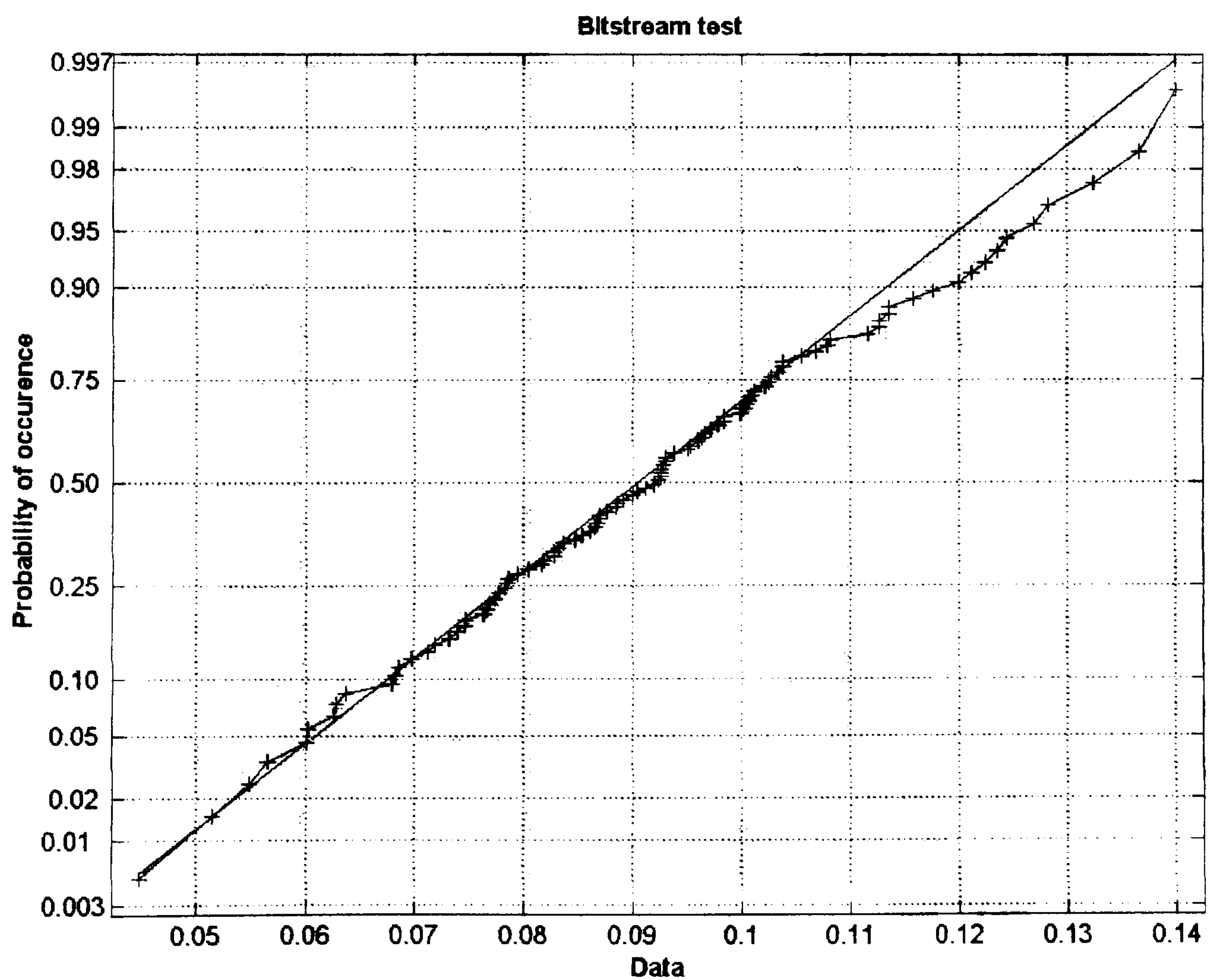


Figure B.6.1: Bitstream test normal pdf – first MWCG, first test

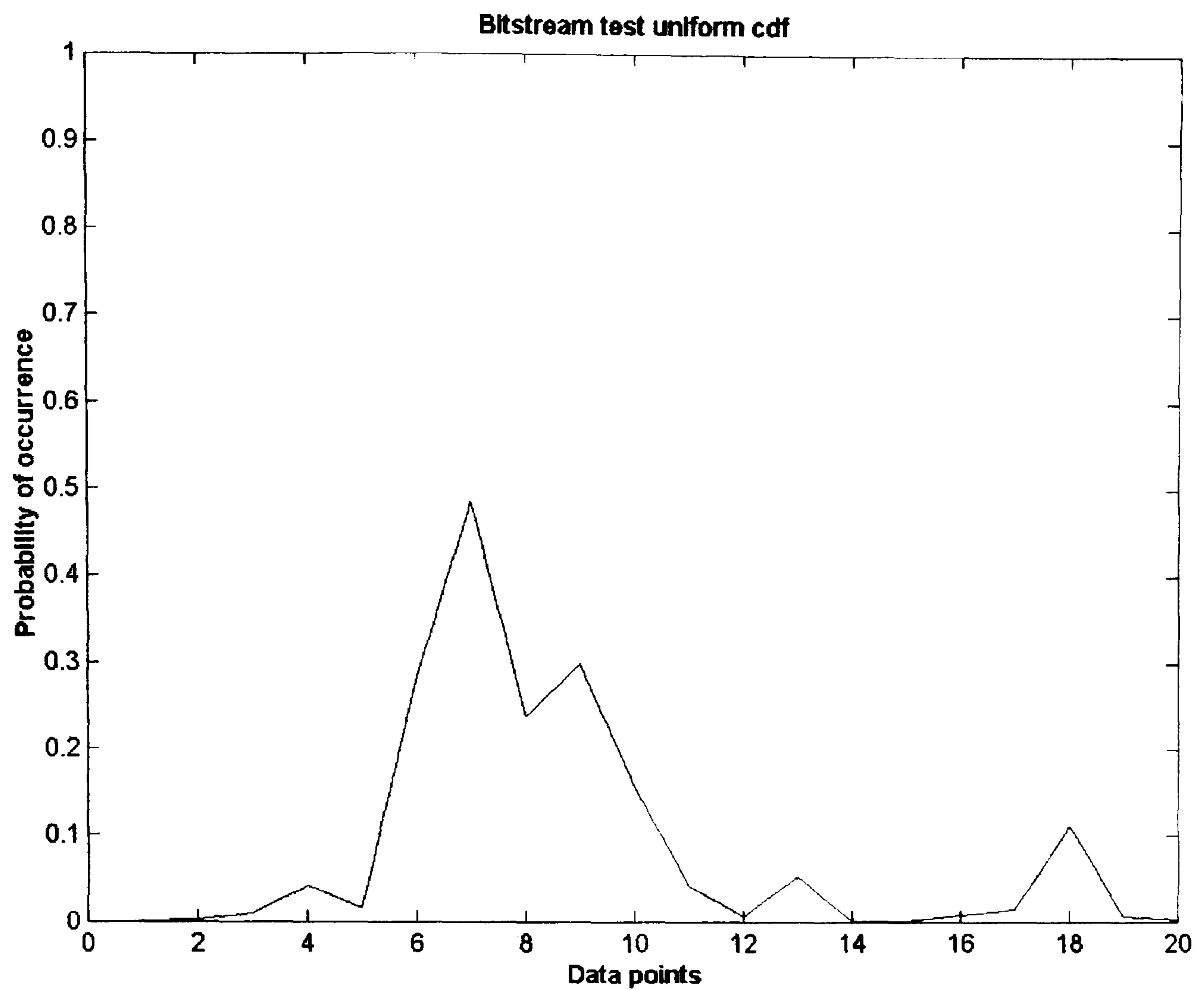


Figure B.6.2: Bitstream test uniform cdf – first MWCG, first test

B.1.2 MWCG of Equation 5-1 – 2nd Test Set

Seed values: 757, 12941, 64153

Carry values: 13, 10, 701

Multipliers: $a1 = 2111111111$, $a2 = 2131995753$, $a3 = 1517746329$, $b1 = 1447497129$, $b2 = 24456079$, $b3 = 94901263$

Key obtained: 3788971b 9291d7fe aa8a454e 66ff402c ae7bb7f6

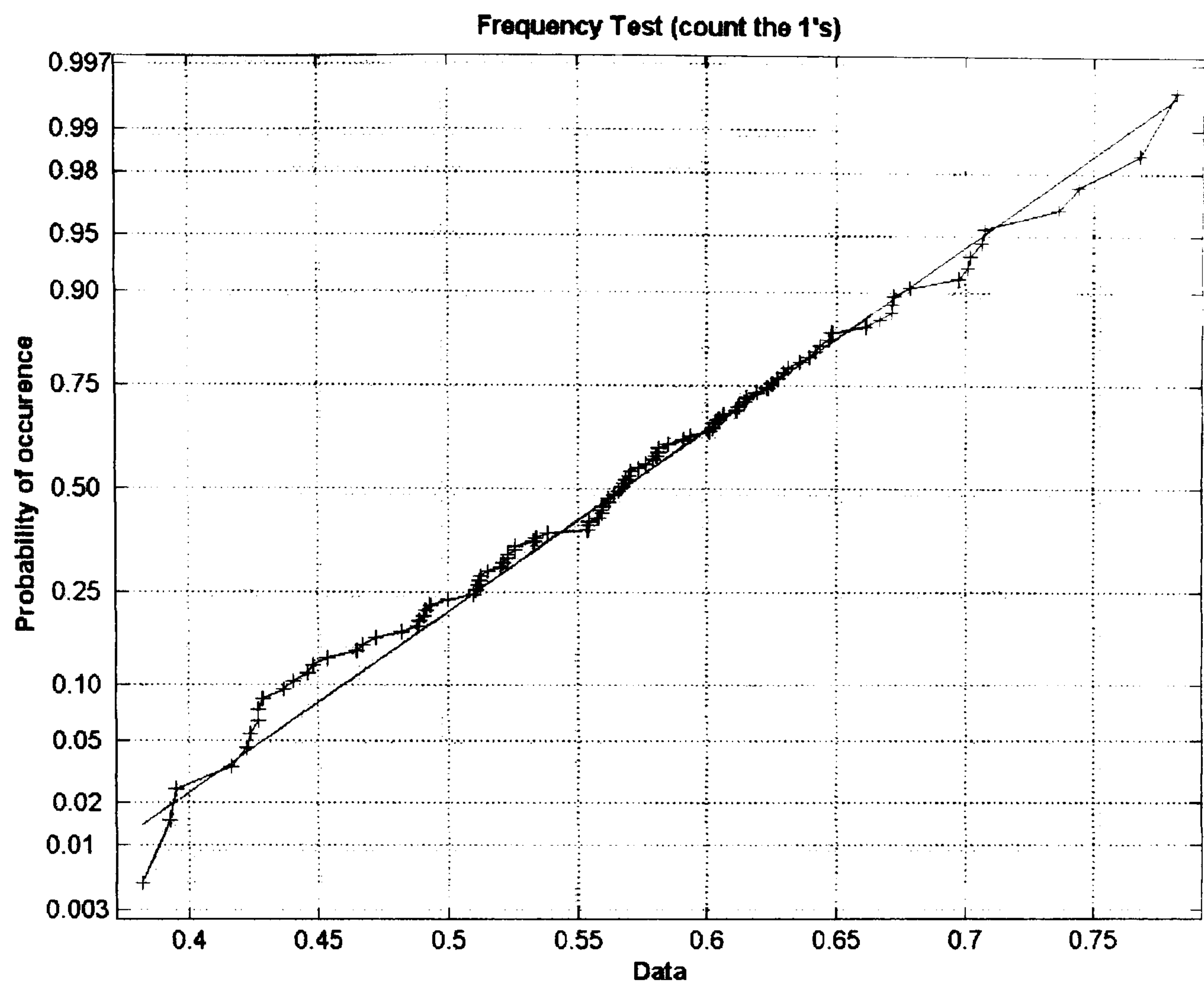


Figure B.7.1: Frequency test normal pdf – first MWCG, second test

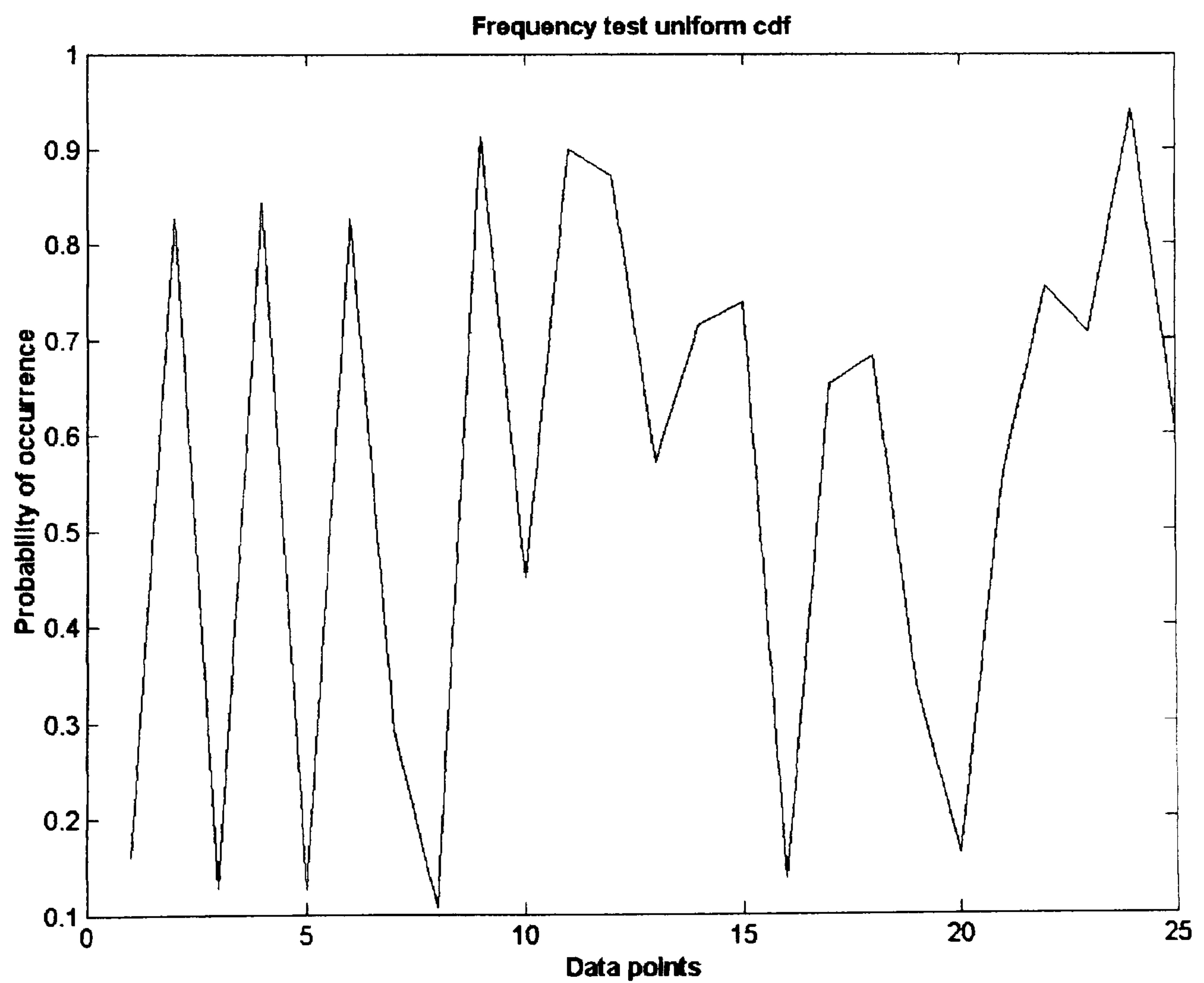


Figure B.7.2: Frequency test uniform cdf – first MWCG, second test

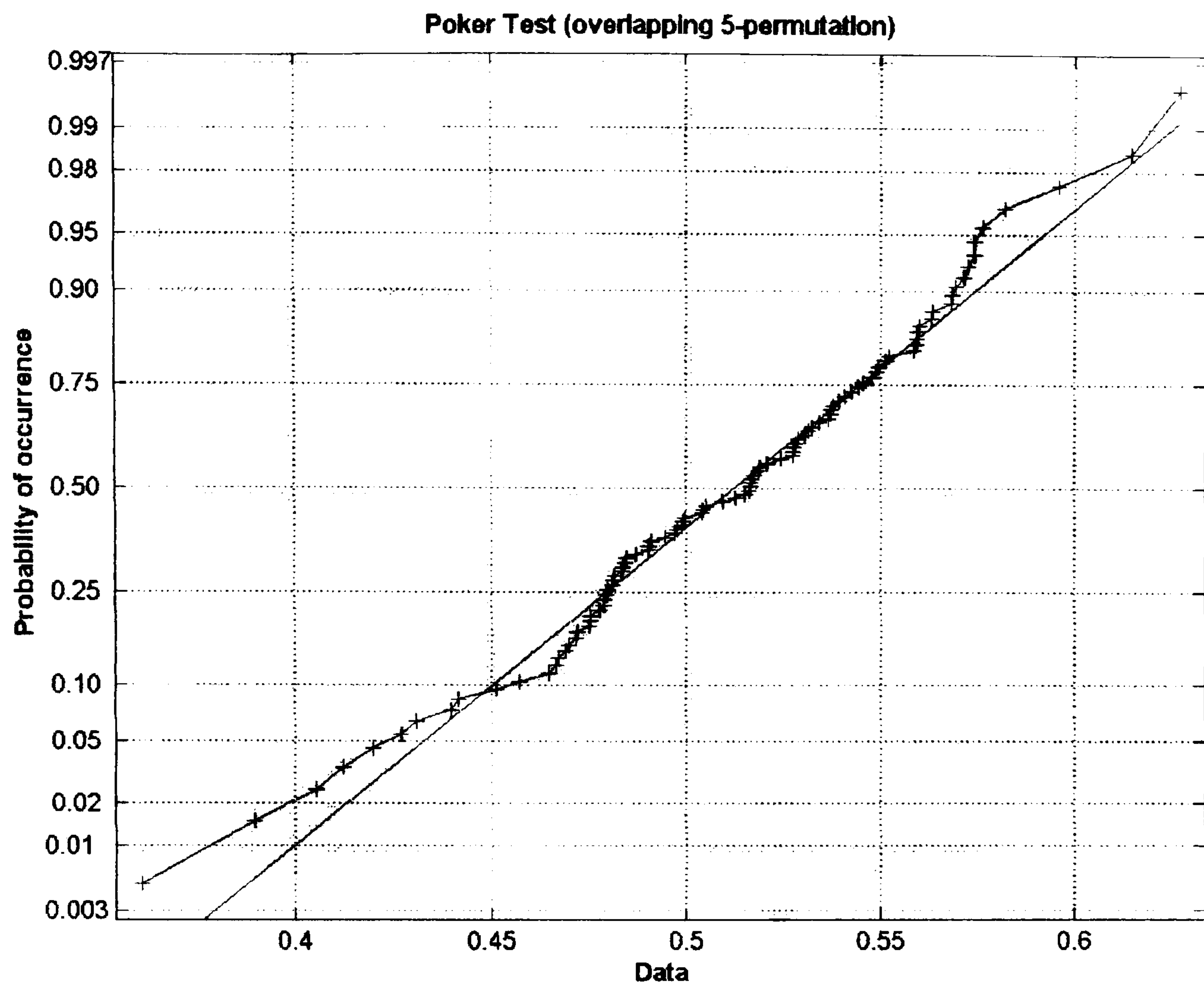


Figure B.8.1: Poker test normal pdf – first MWCG, second test

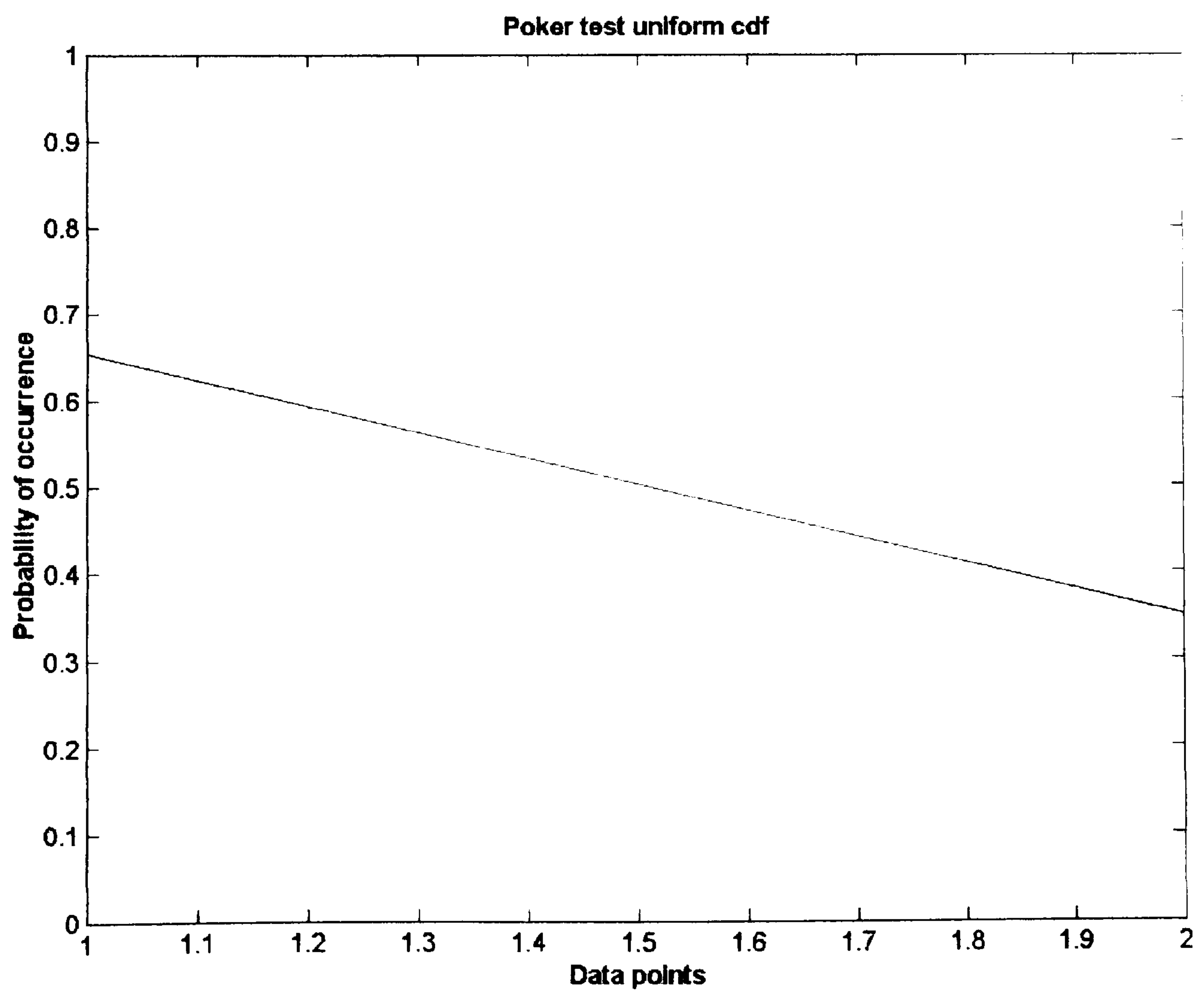


Figure B.8.2: Poker test uniform cdf – first MWCG, second test

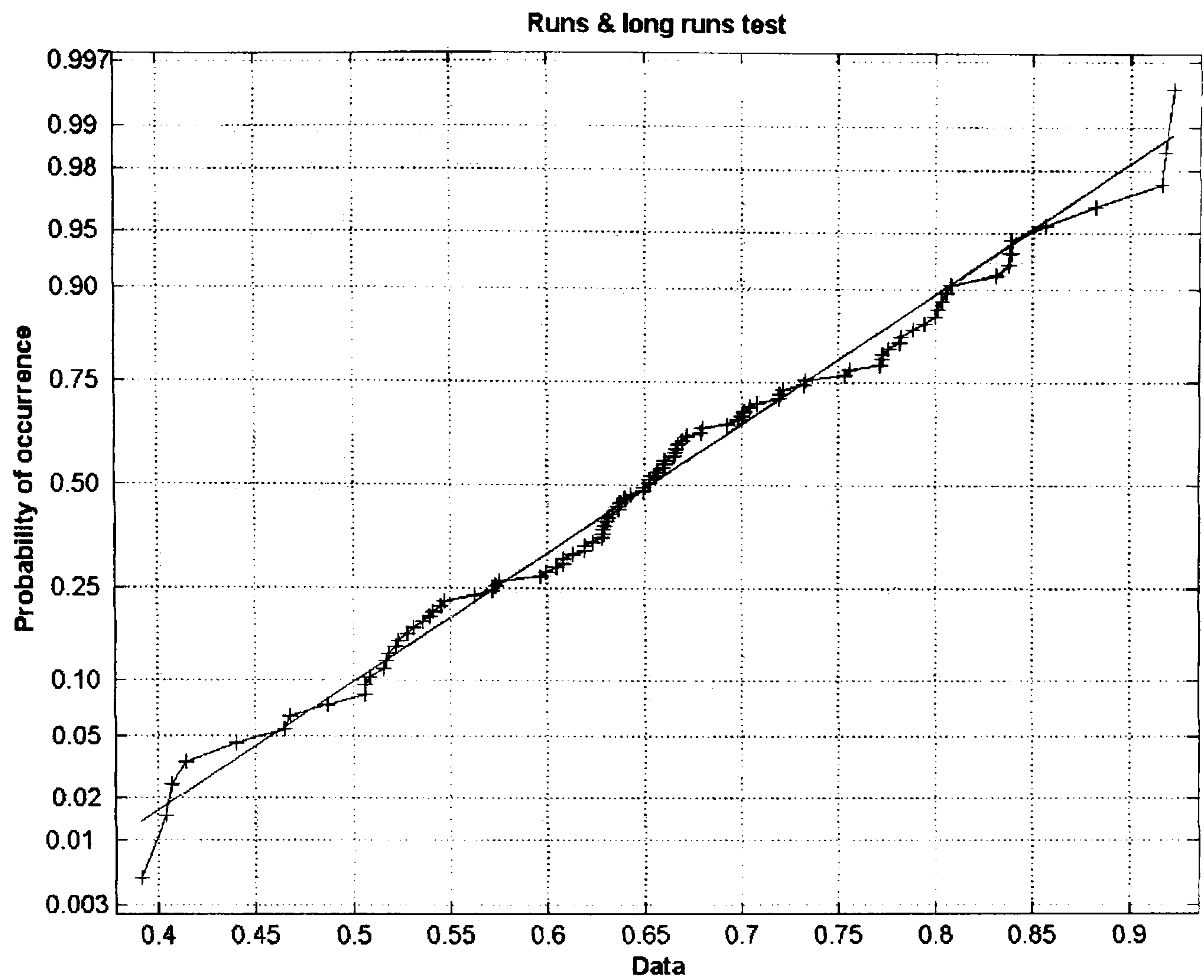


Figure B.9.1: Runs and long runs tests normal pdf – first MWCG, second test

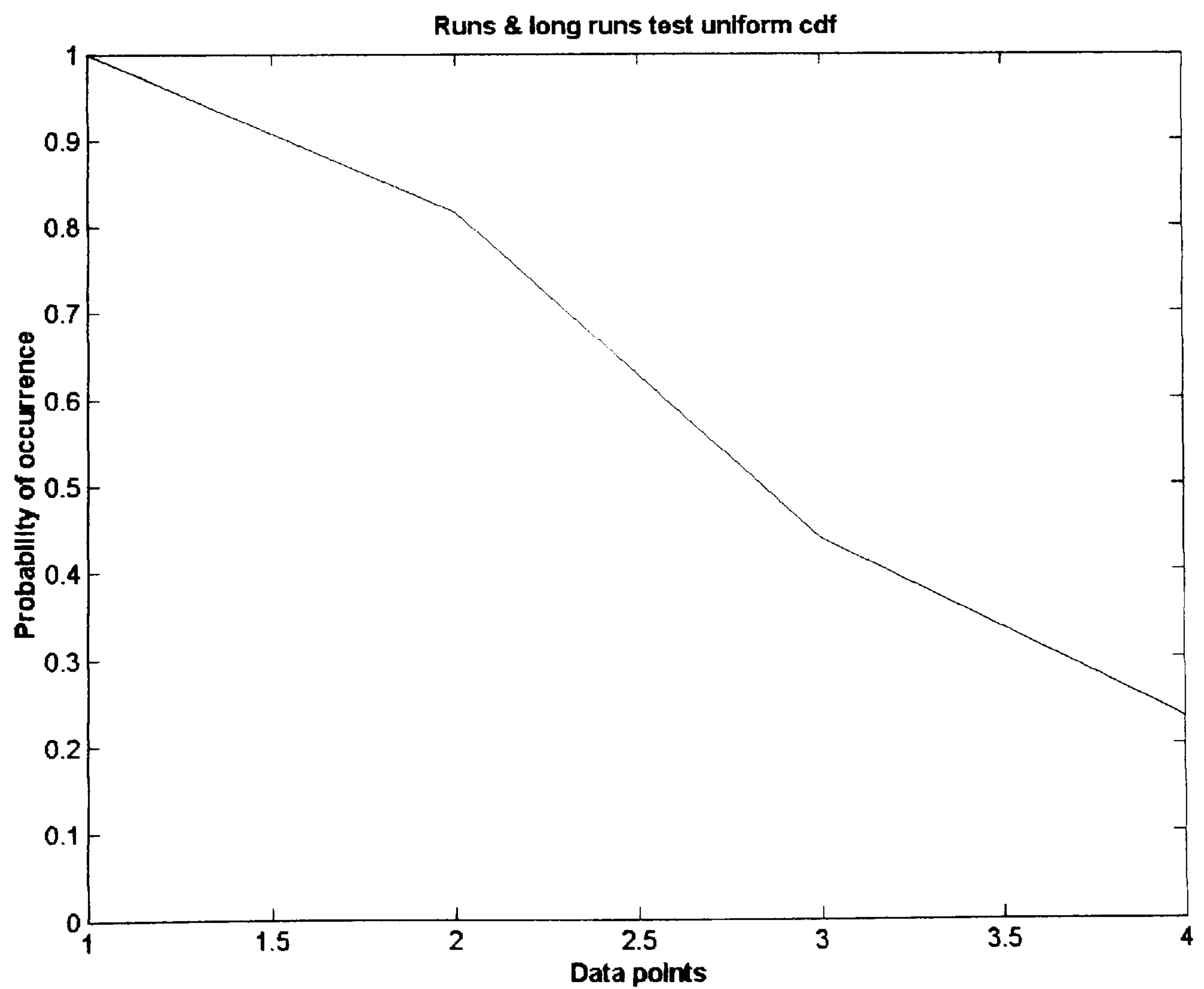


Figure B.9.2: Runs and long runs test uniform cdf – first MWCG, second test

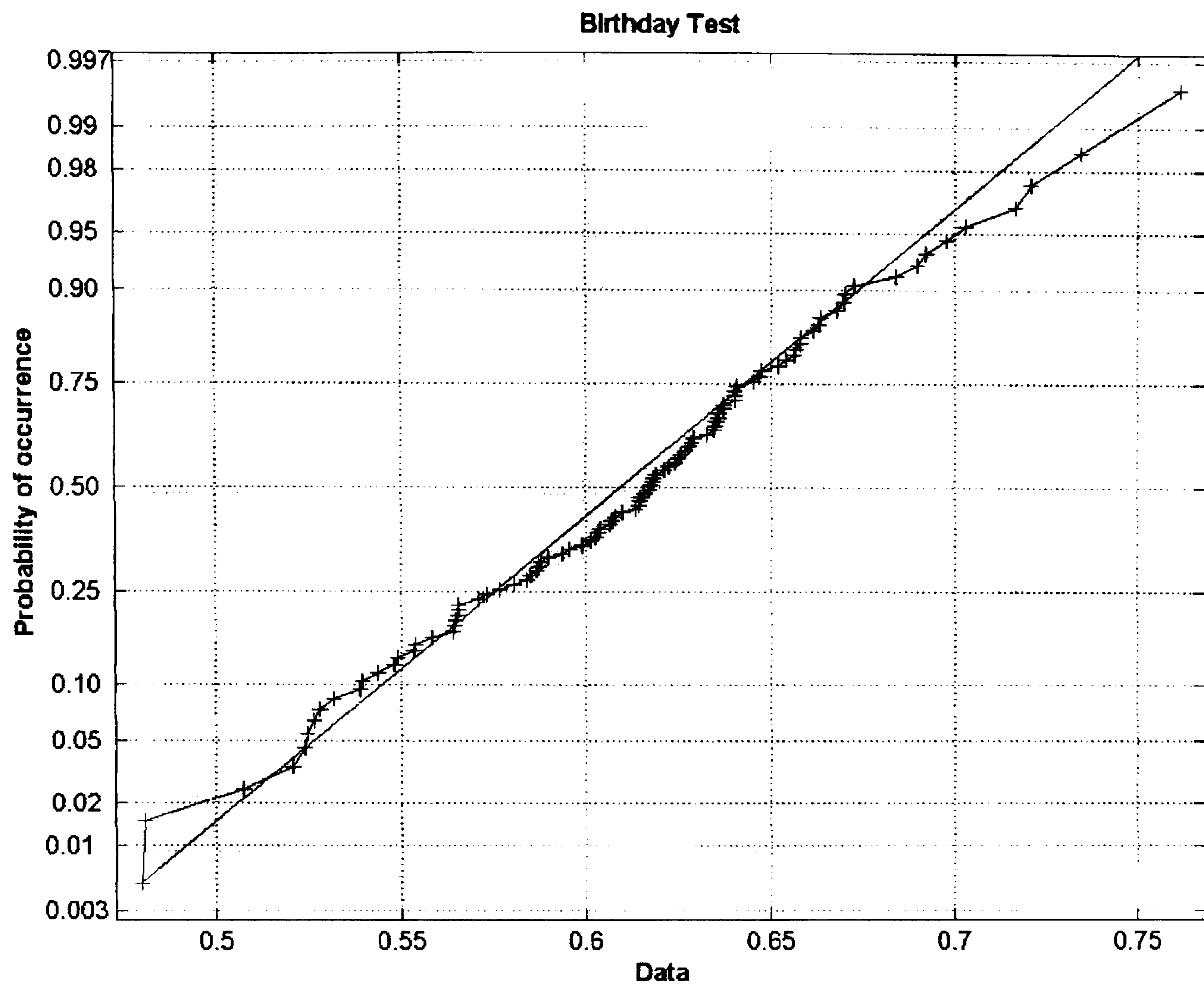


Figure B.10.1: Birthday test normal pdf – first MWCG, second test

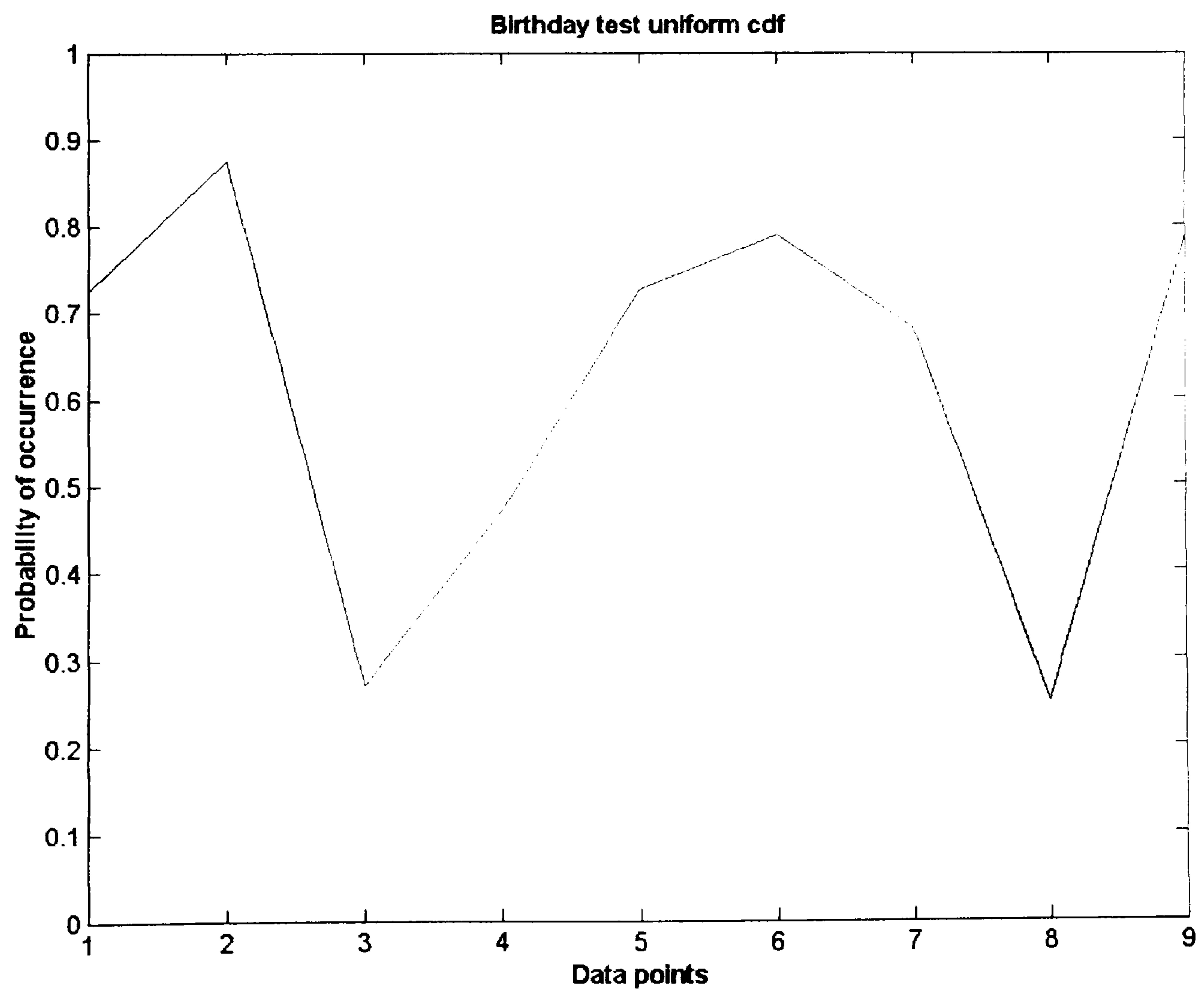


Figure B.10.2: Birthday test uniform cdf – first MWCG, second test

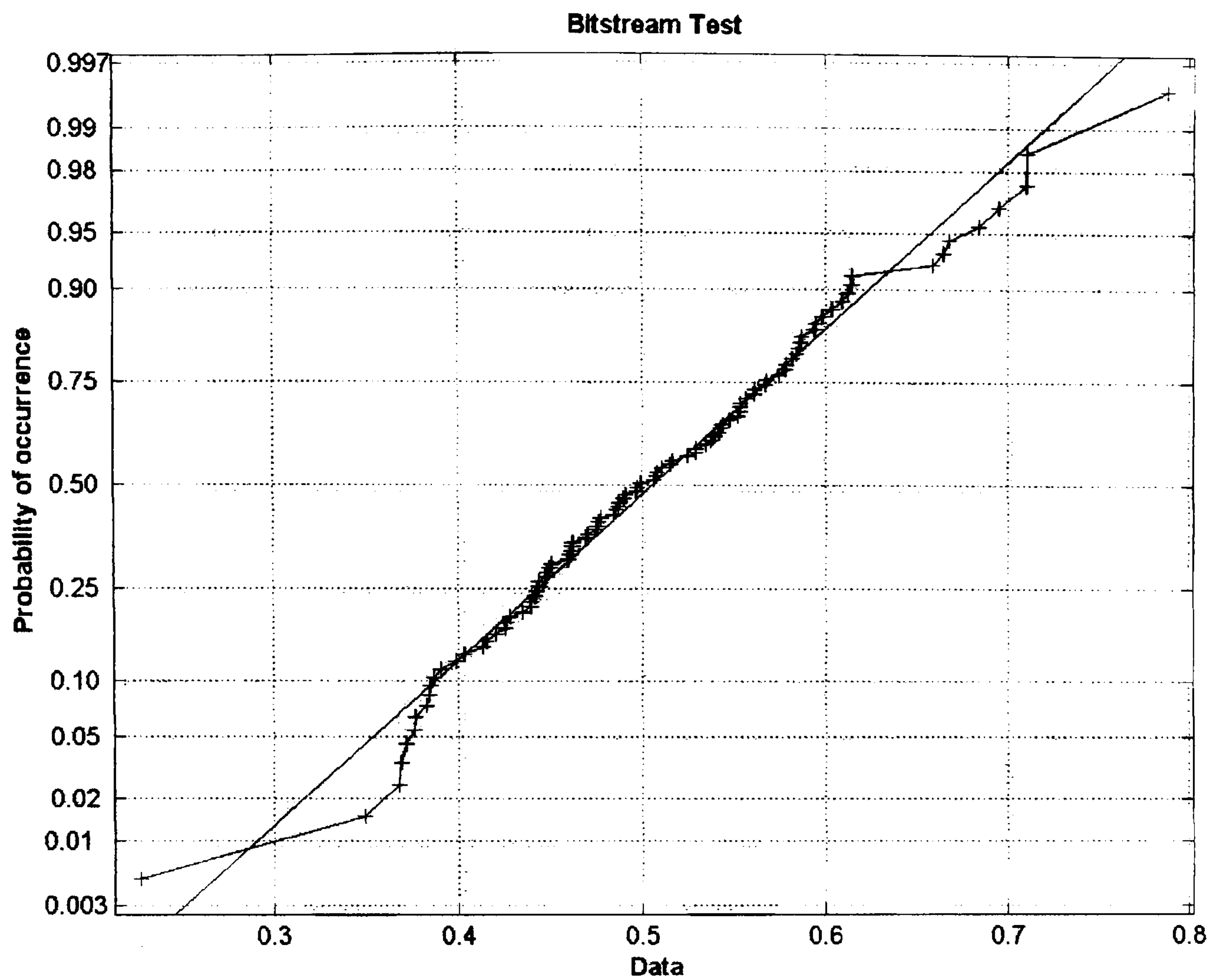


Figure B.11.1: Bitstream test normal pdf – first MWCG, second test

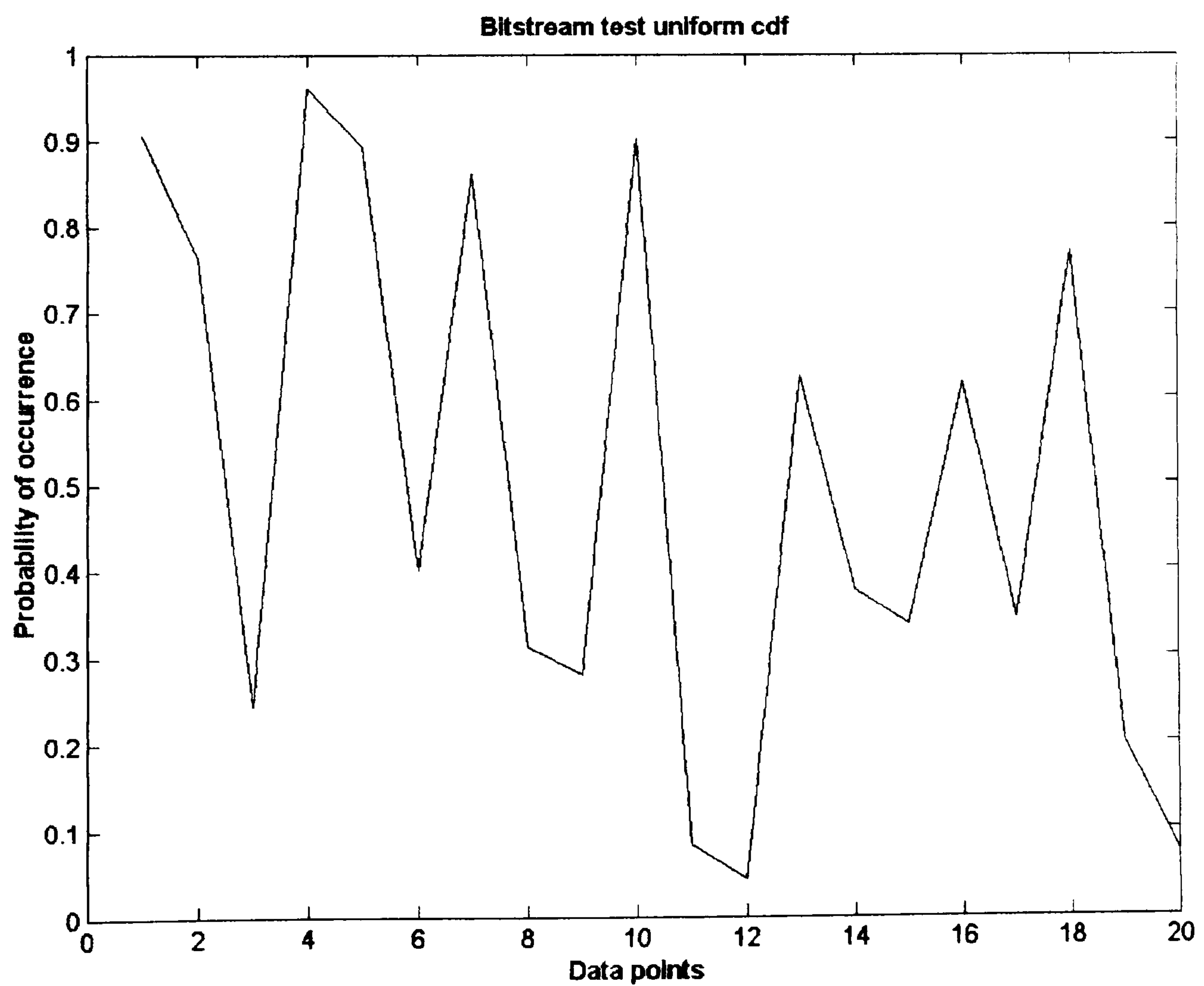


Figure B.11.2: Bitstream test uniform cdf – first MWCG, second test

B.1.3 MWCG of Equation 5-2 – 1st Test Set

Seed values: 9941, 7951, 12011

Carry values: 12007, 58337, 56003

Multipliers: $a_1 = 1517746329$, $a_2 = 1557985959$, $a_3 = 1447497129$, $b_1 = 24456079$, $b_2 = 48897674$, $b_3 = 94901263$

Key obtained: f029baa4 39276c4a e2687986 8c429a49 5258724a

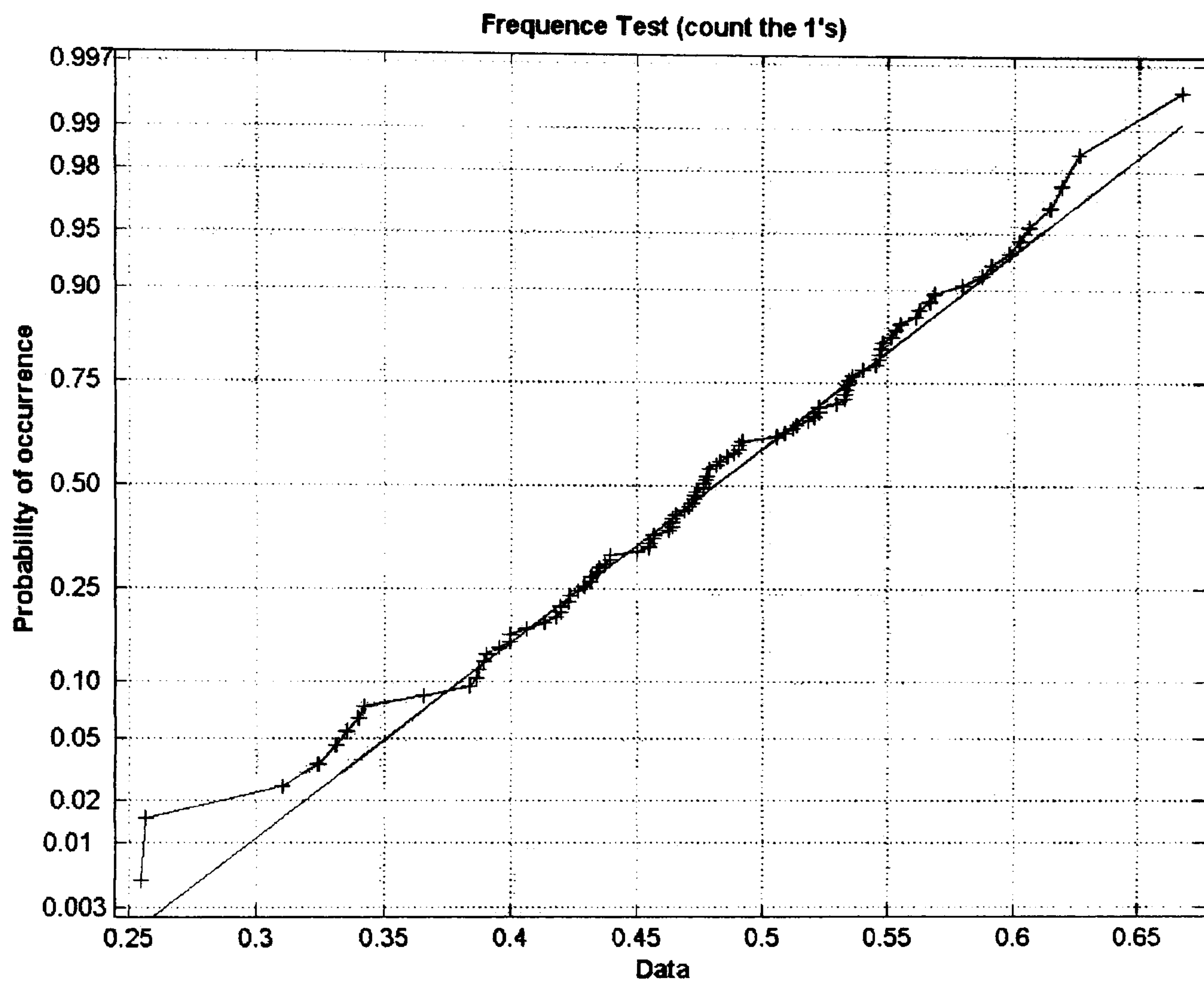


Figure B.12.1: Frequency test normal pdf – second MWCG, first test

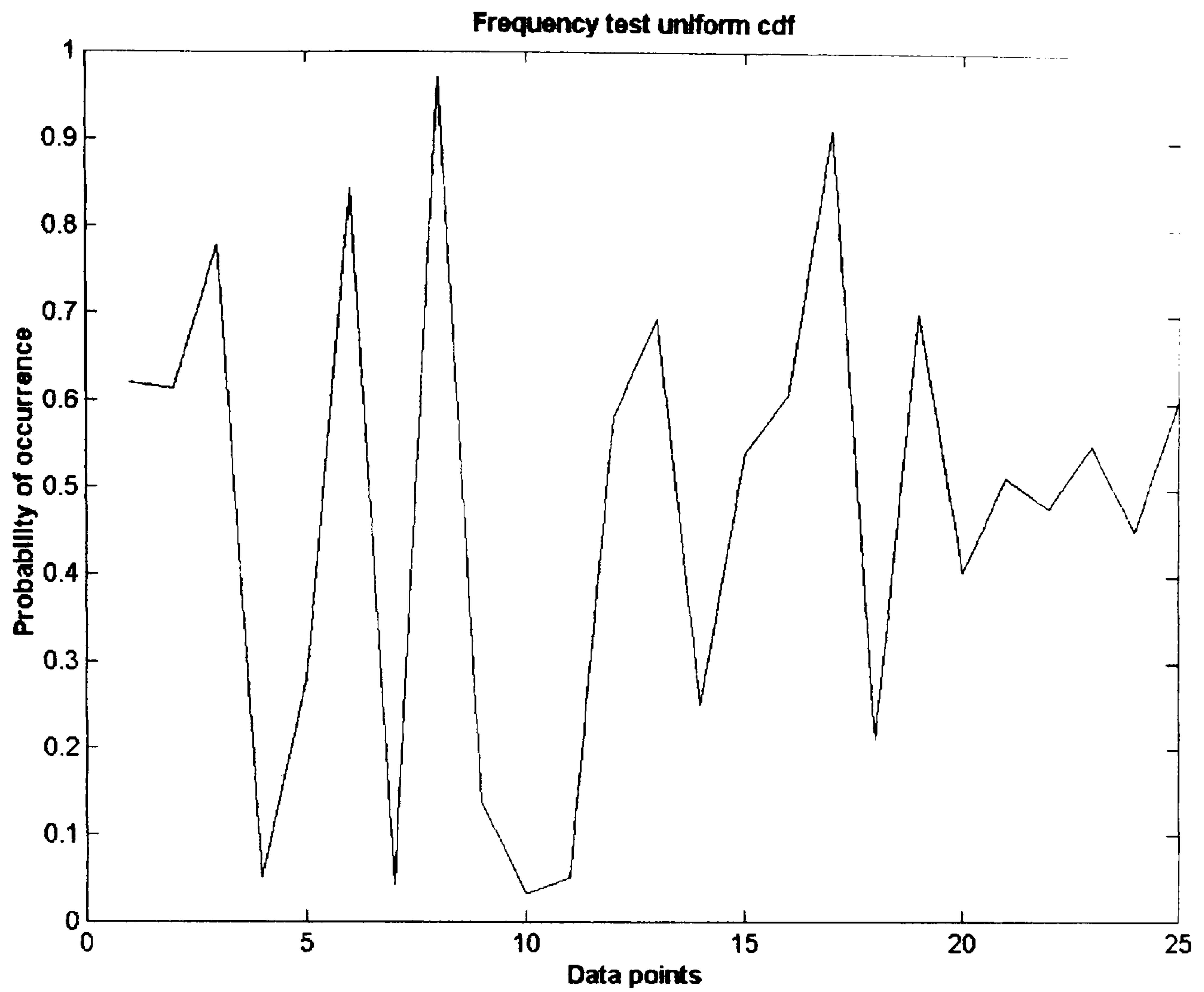


Figure B.12.2: Frequency test uniform cdf – second MWCG, first test

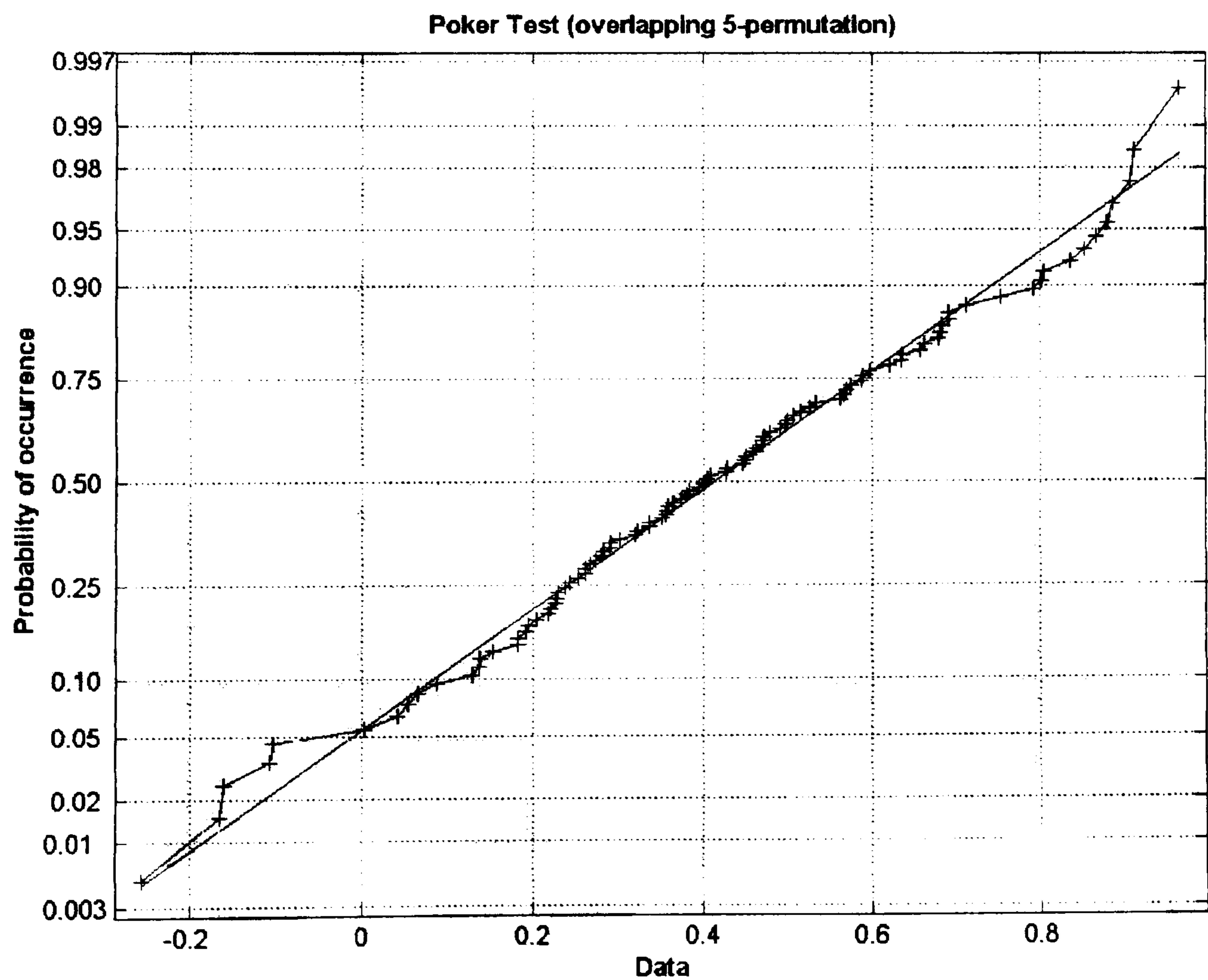


Figure B.13.1: Poker test normal pdf – second MWCG, first test

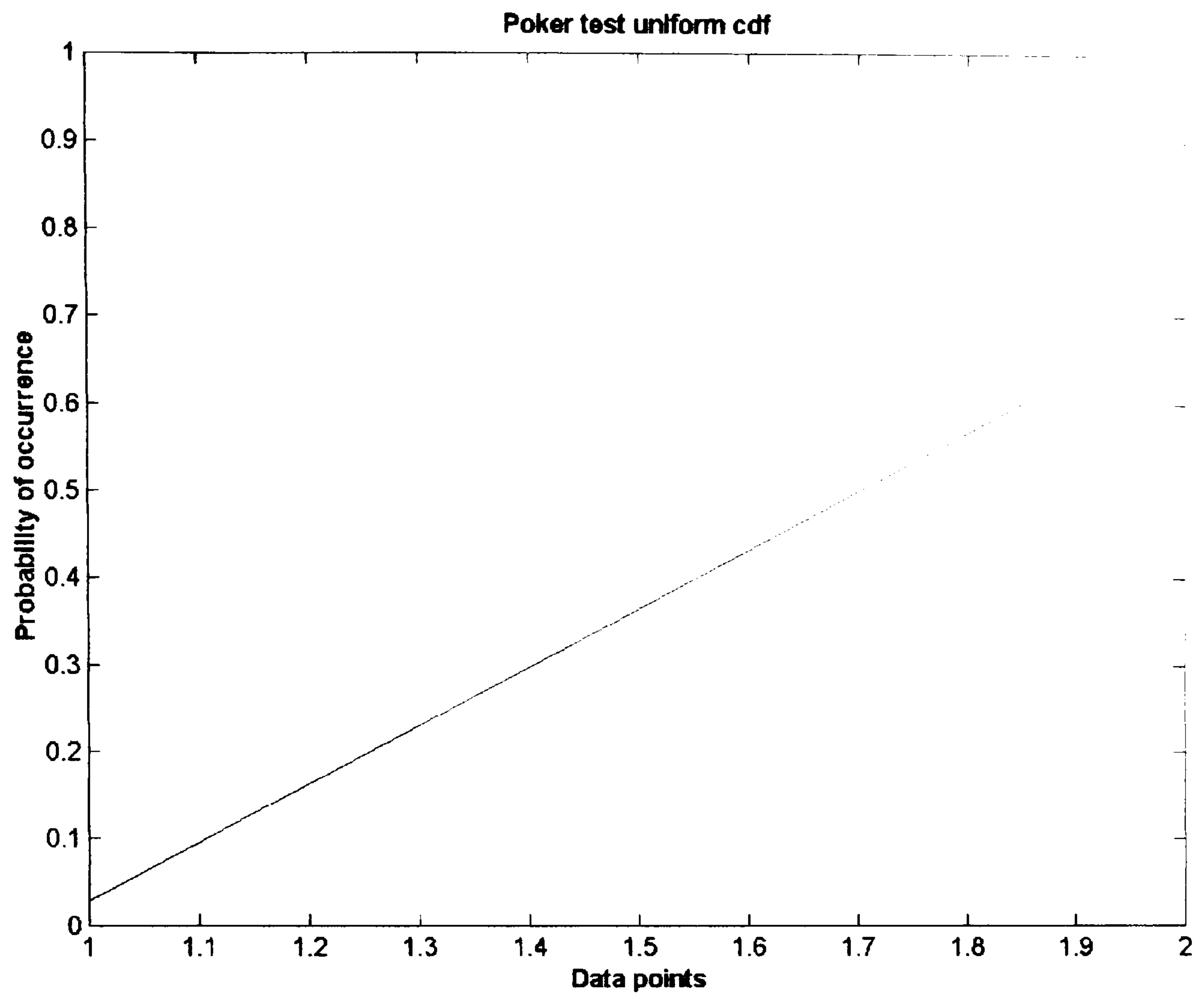


Figure B.13.2: Poker test uniform cdf – second MWCG, first test

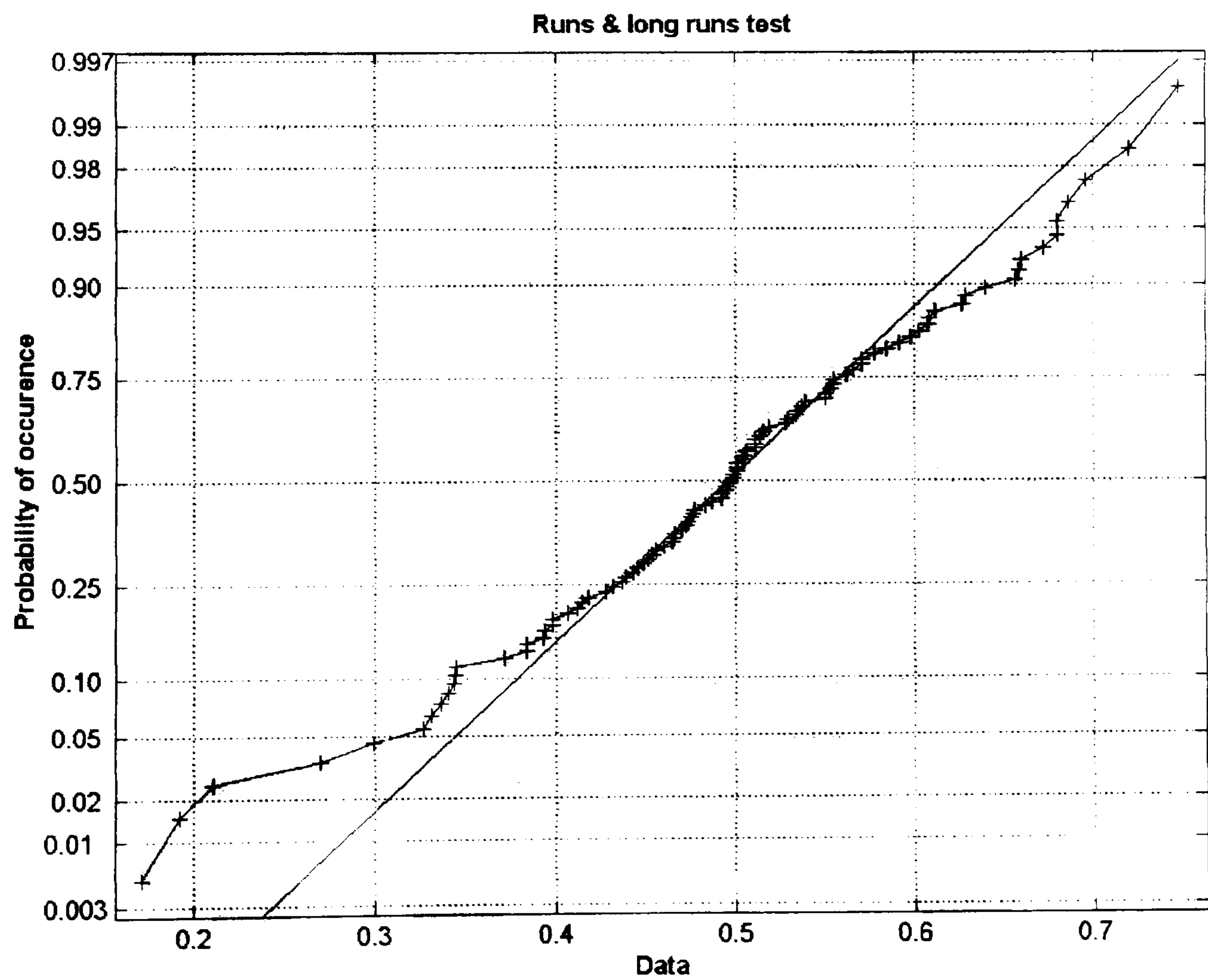


Figure B.14.1: Runs and long runs tests normal pdf – second MWCG, first test

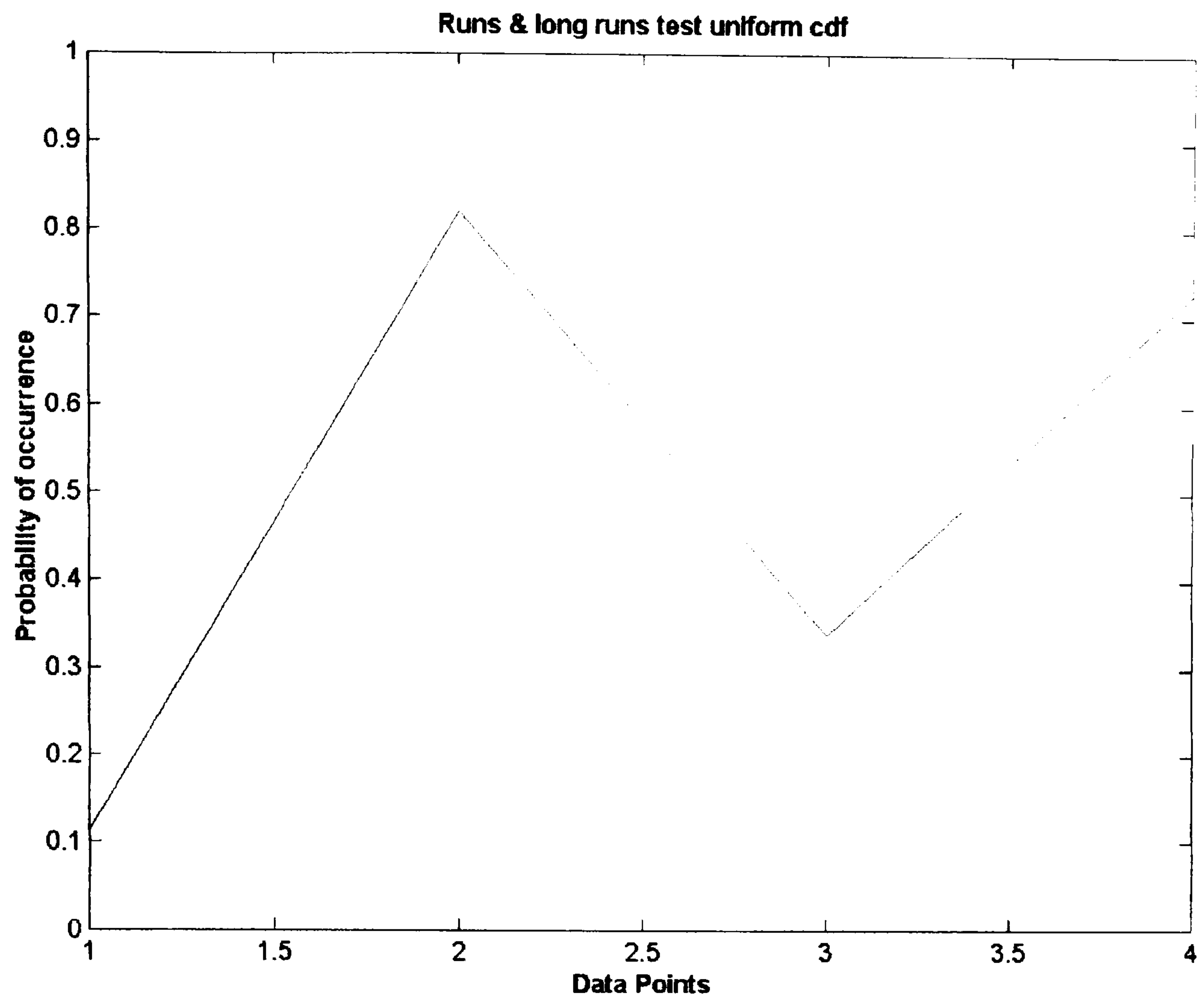


Figure B.14.2: Runs and long runs test uniform cdf – second MWCG, first test

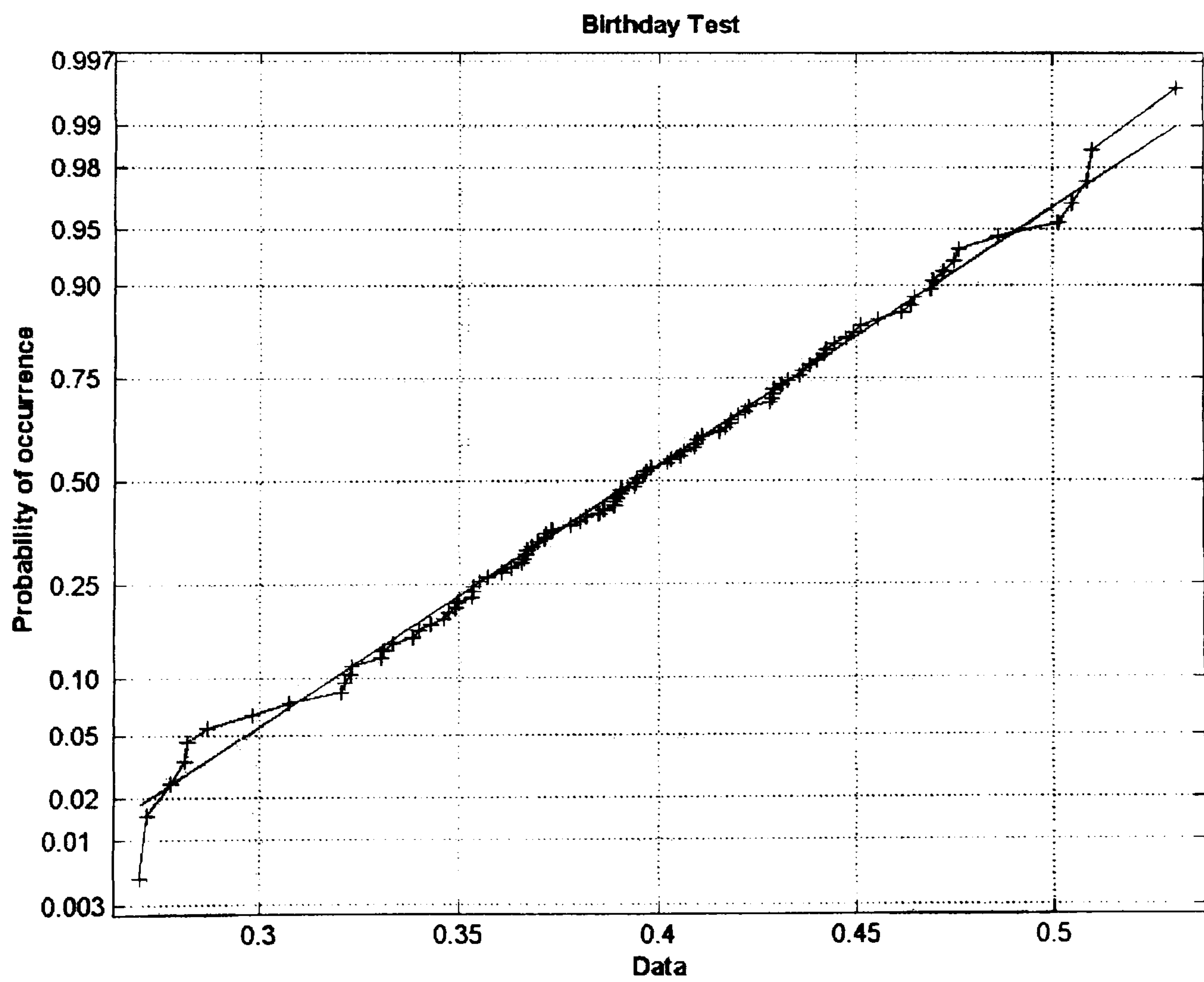


Figure B.15.1: Birthday test normal pdf – second MWCG, first test

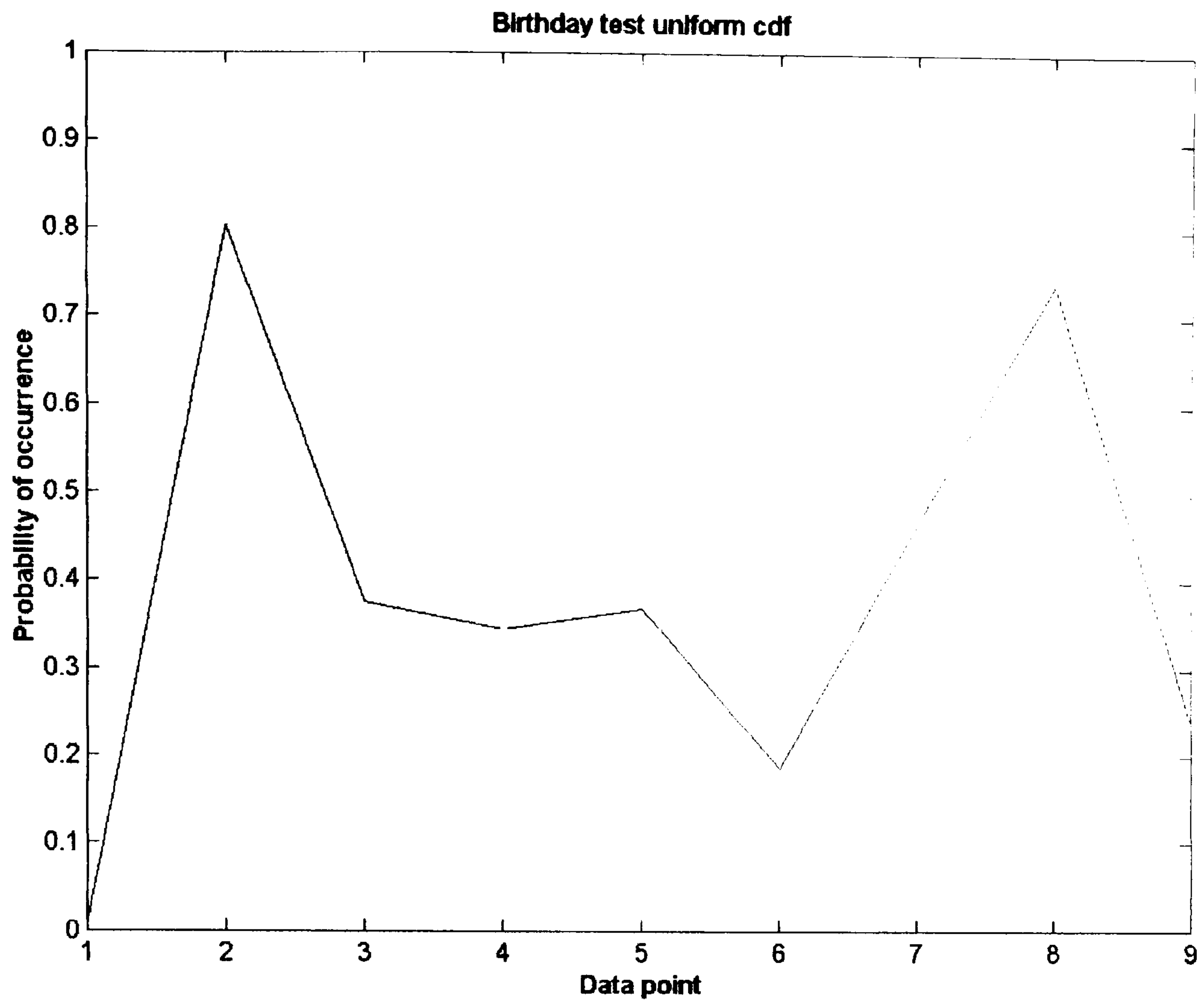


Figure B.15.2: Birthday test uniform cdf – second MWCG, first test

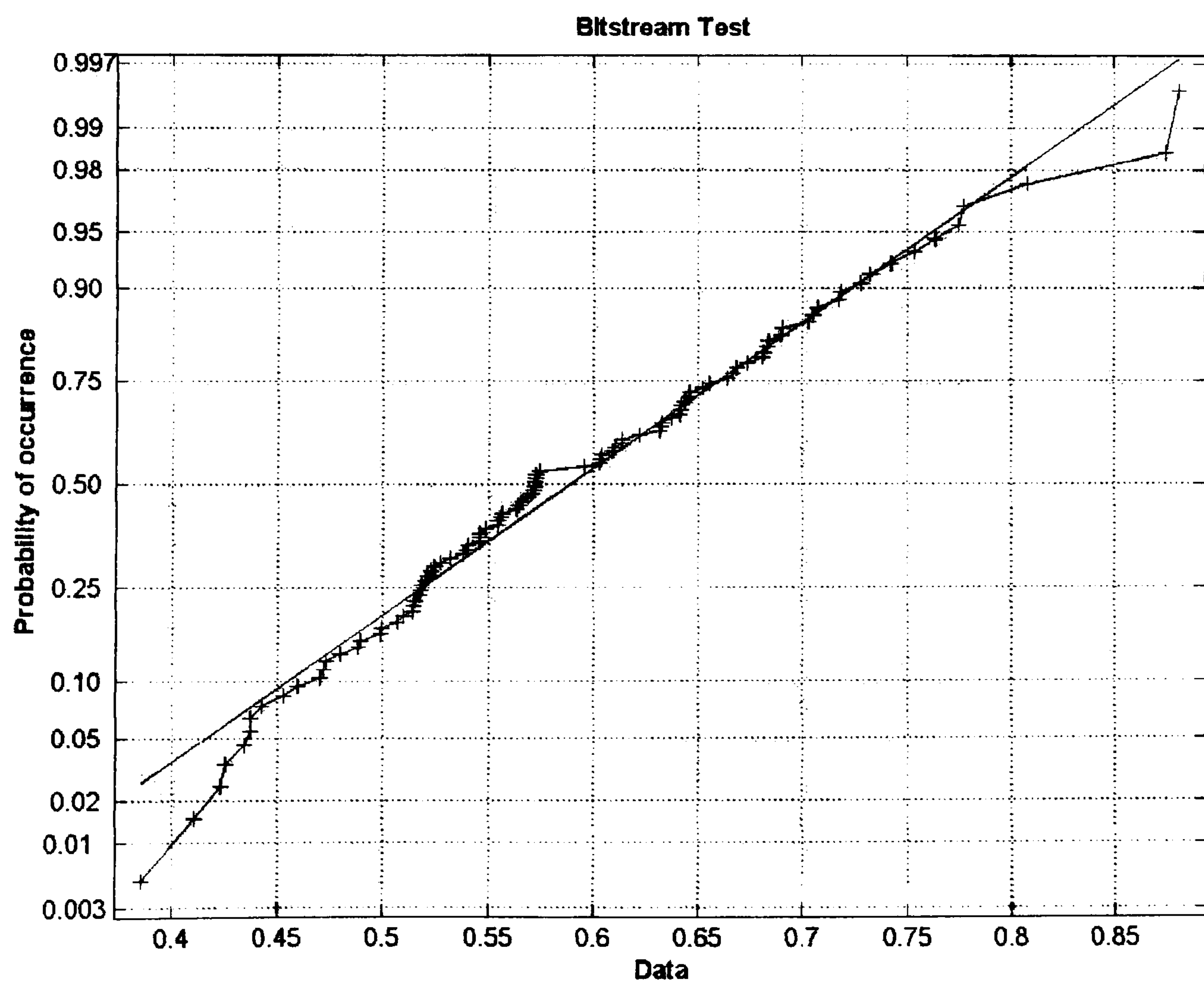


Figure B.16.1: Bitstream test normal pdf – second MWCG, first test

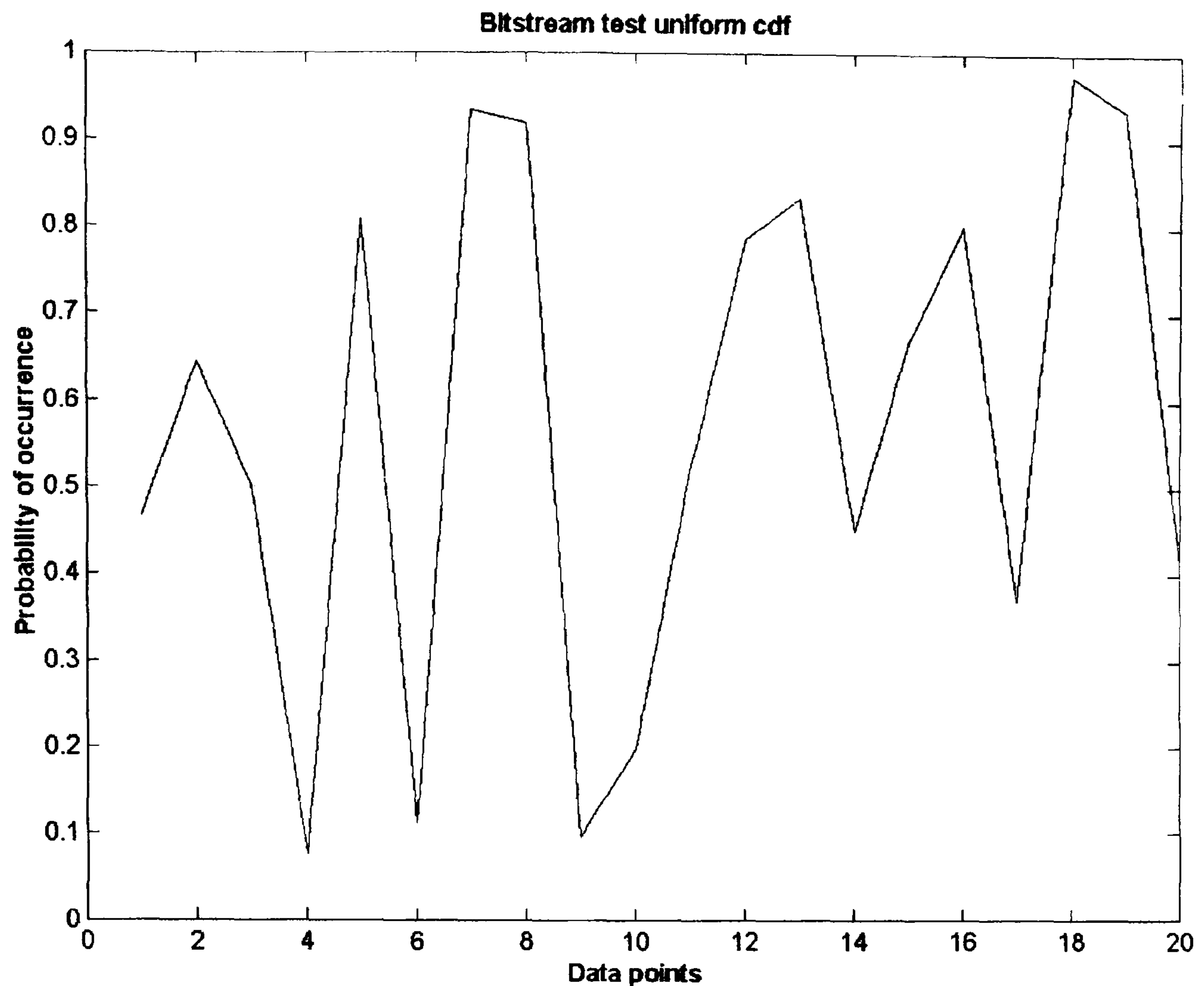


Figure B.16.2: Bitstream test uniform cdf – second MWCG, first test

B.1.4 MWCG of Equation 5-2 – 2nd Test Set

Seed values: 1063, 1171, 3109

Carry values: 17, 23, 31

Multipliers: $a_1 = 1517746329$, $a_2 = 1557985959$, $a_3 = 1447497129$, $b_1 = 24456079$, $b_2 = 48897674$, $b_3 = 94901263$

Key obtained: 59c5812a ad2f0835 61ab8870 77e0191b c64918be

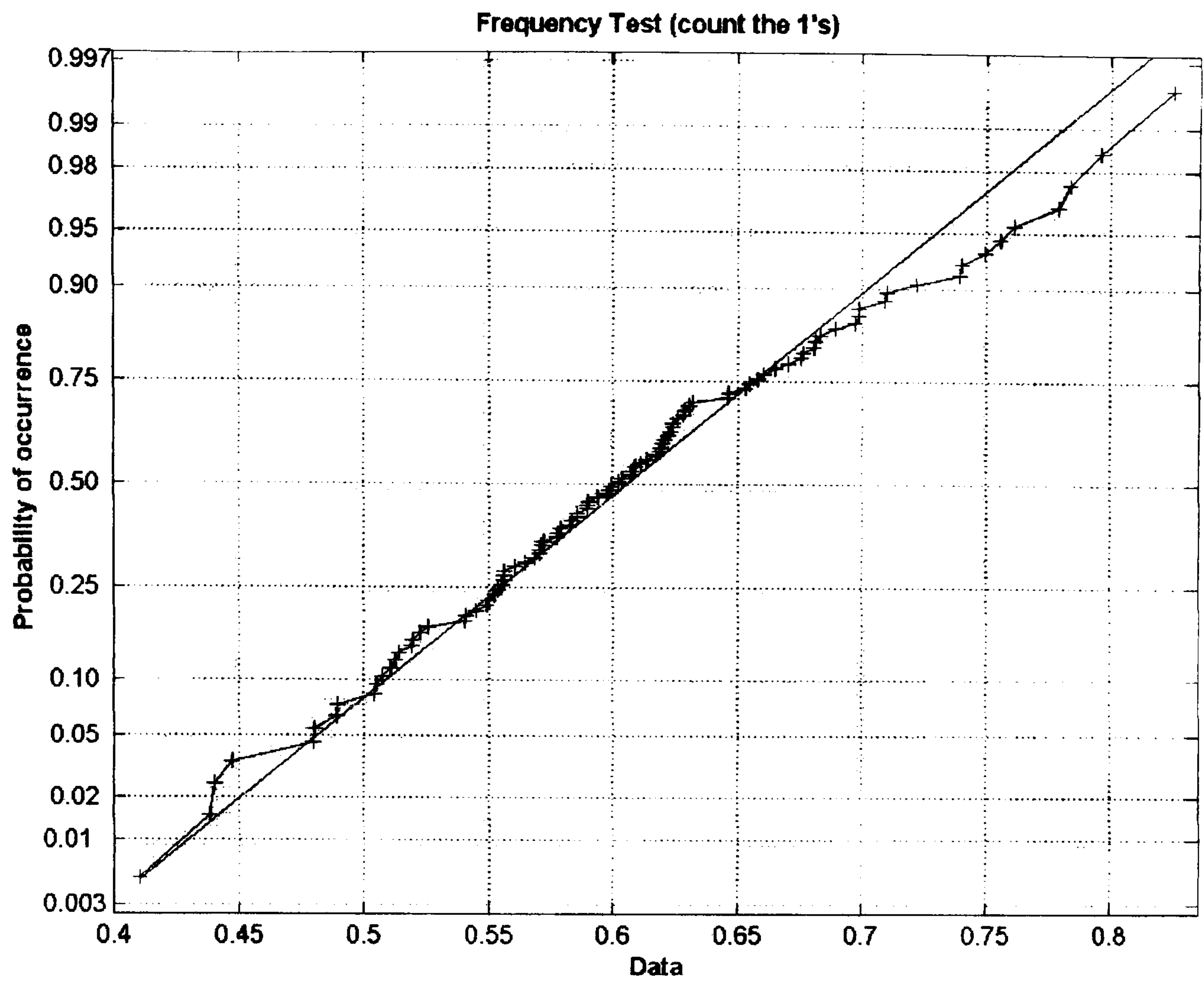


Figure B.17.1: Frequency test normal pdf – second MWCG, second test

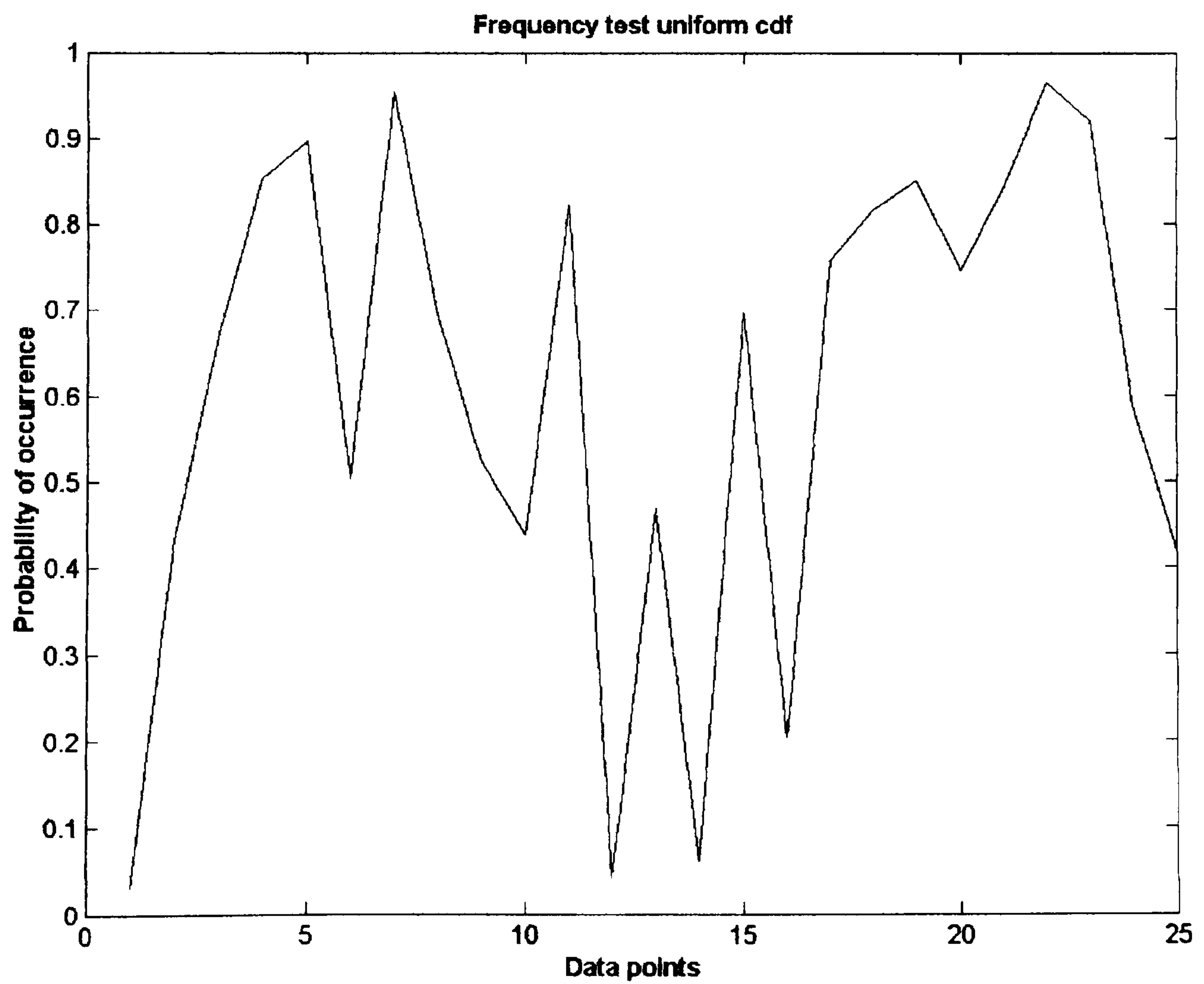


Figure B.17.2: Frequency test uniform cdf – second MWCG, second test

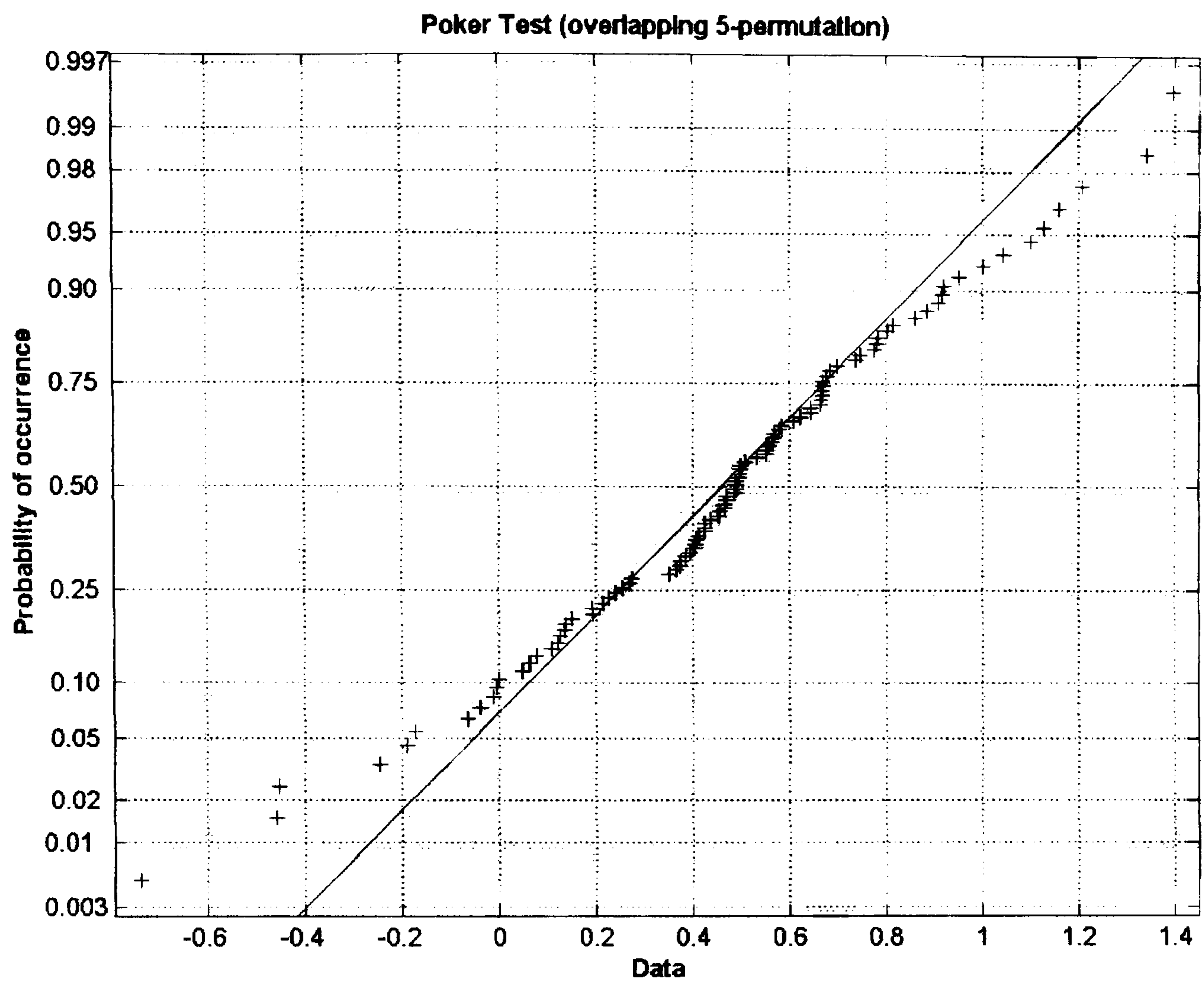


Figure B.18.1: Poker test normal pdf – second MWCG, second test

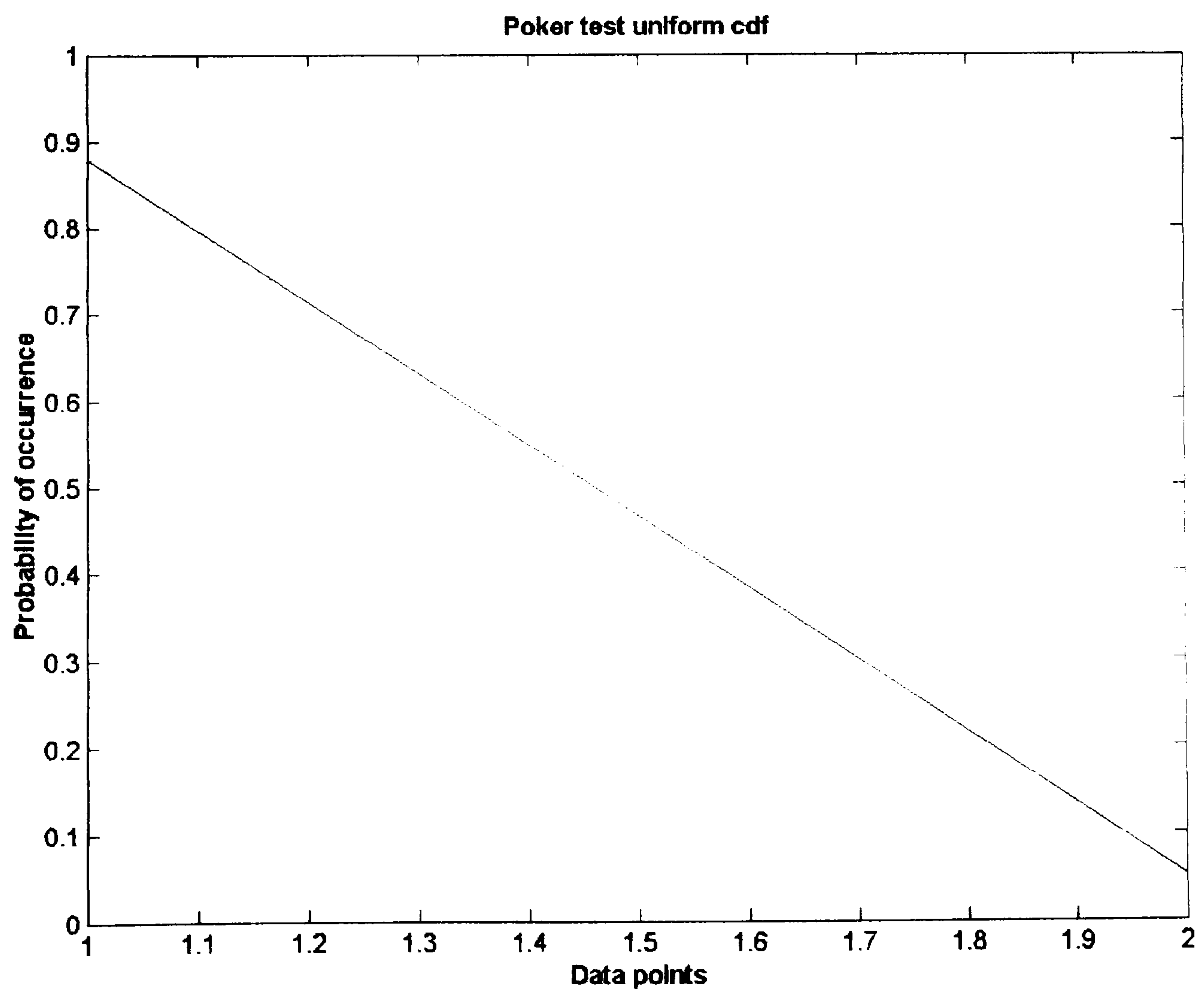


Figure B.18.2: Poker test uniform cdf – second MWCG, second test

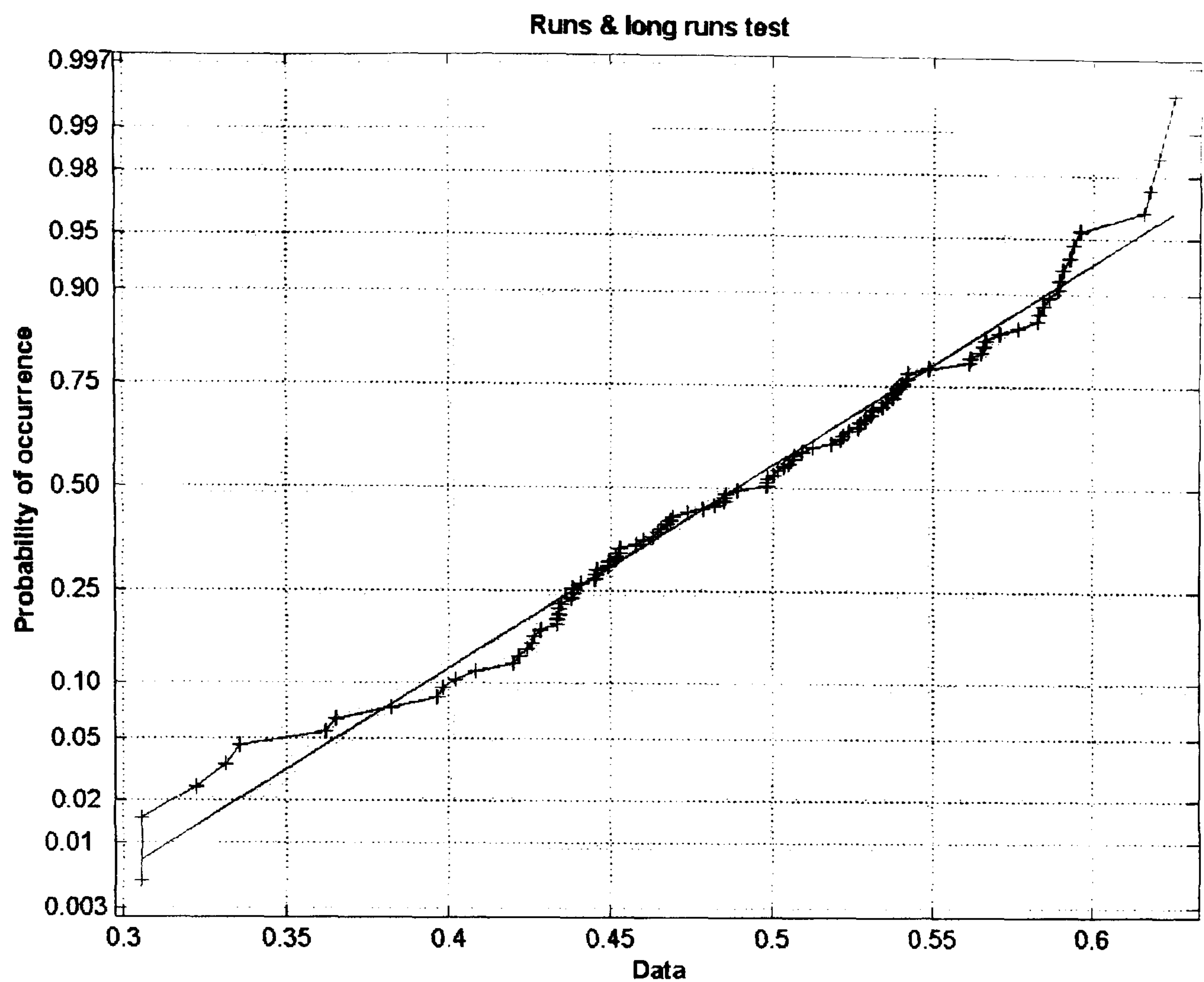


Figure B.19.1: Runs and long runs test normal pdf – second MWCG, second test

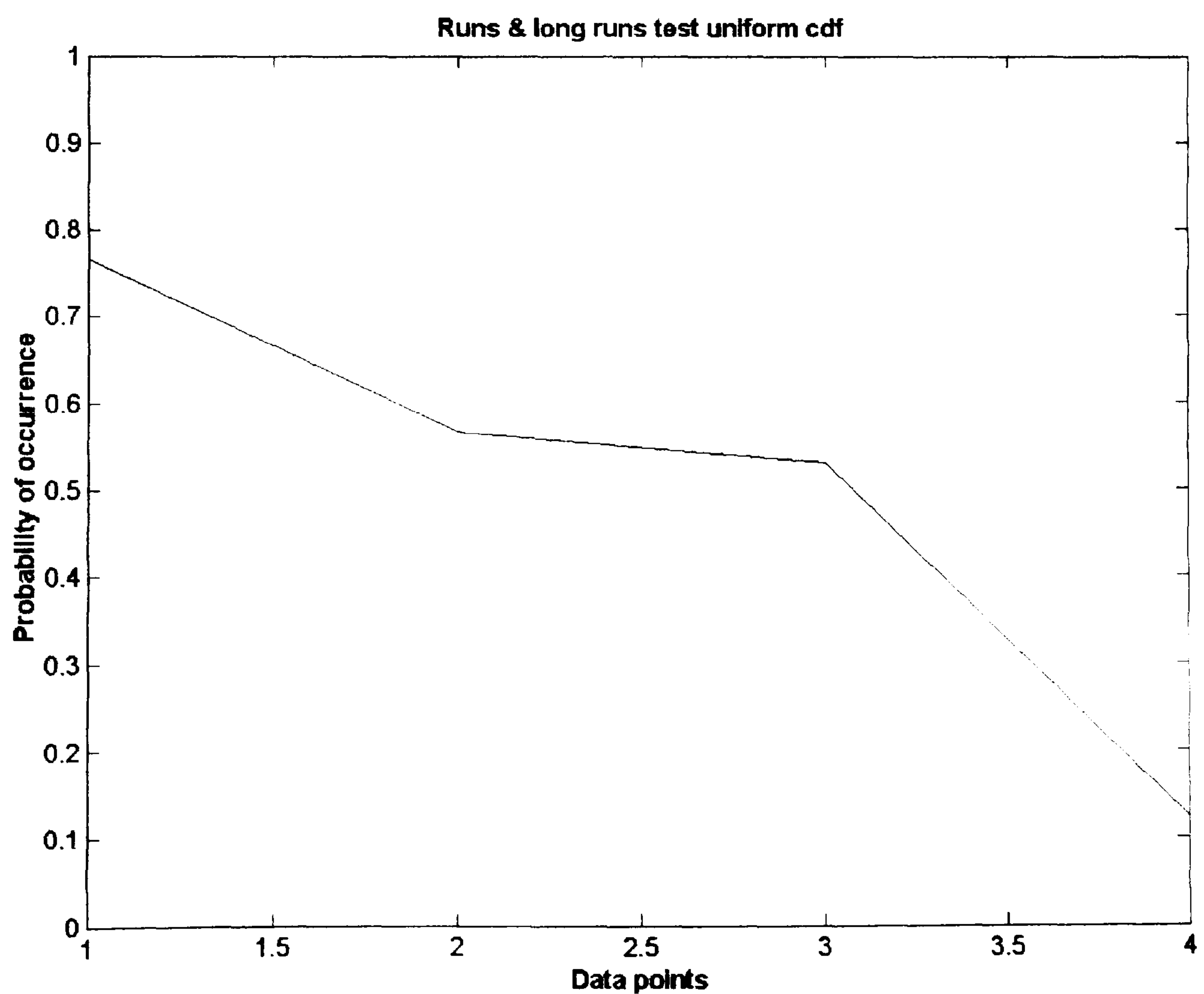


Figure B.19.2: Runs and long runs test uniform cdf – second MWCG, second test

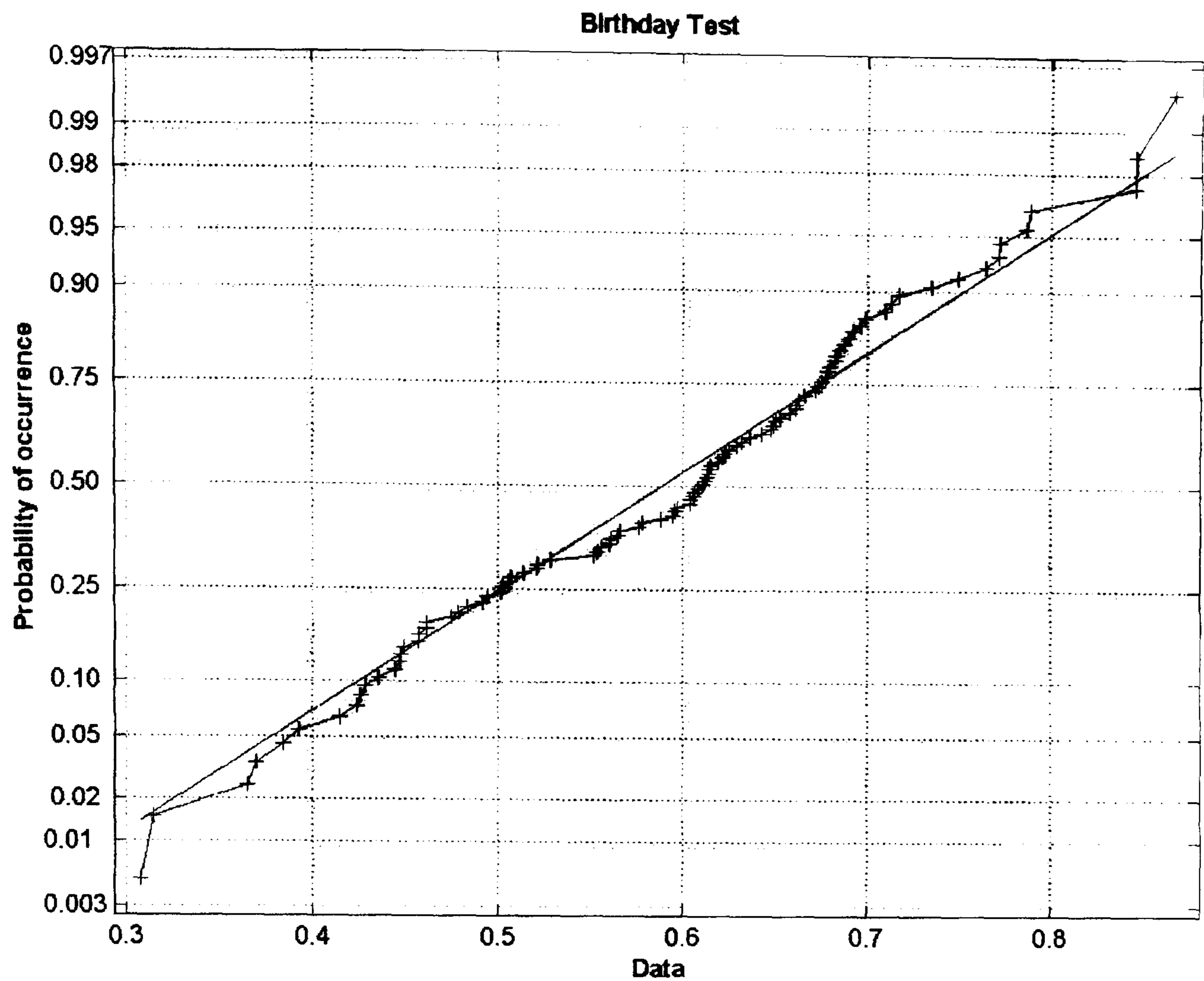


Figure B.20.1: Birthday test normal pdf – second MWCG, second test

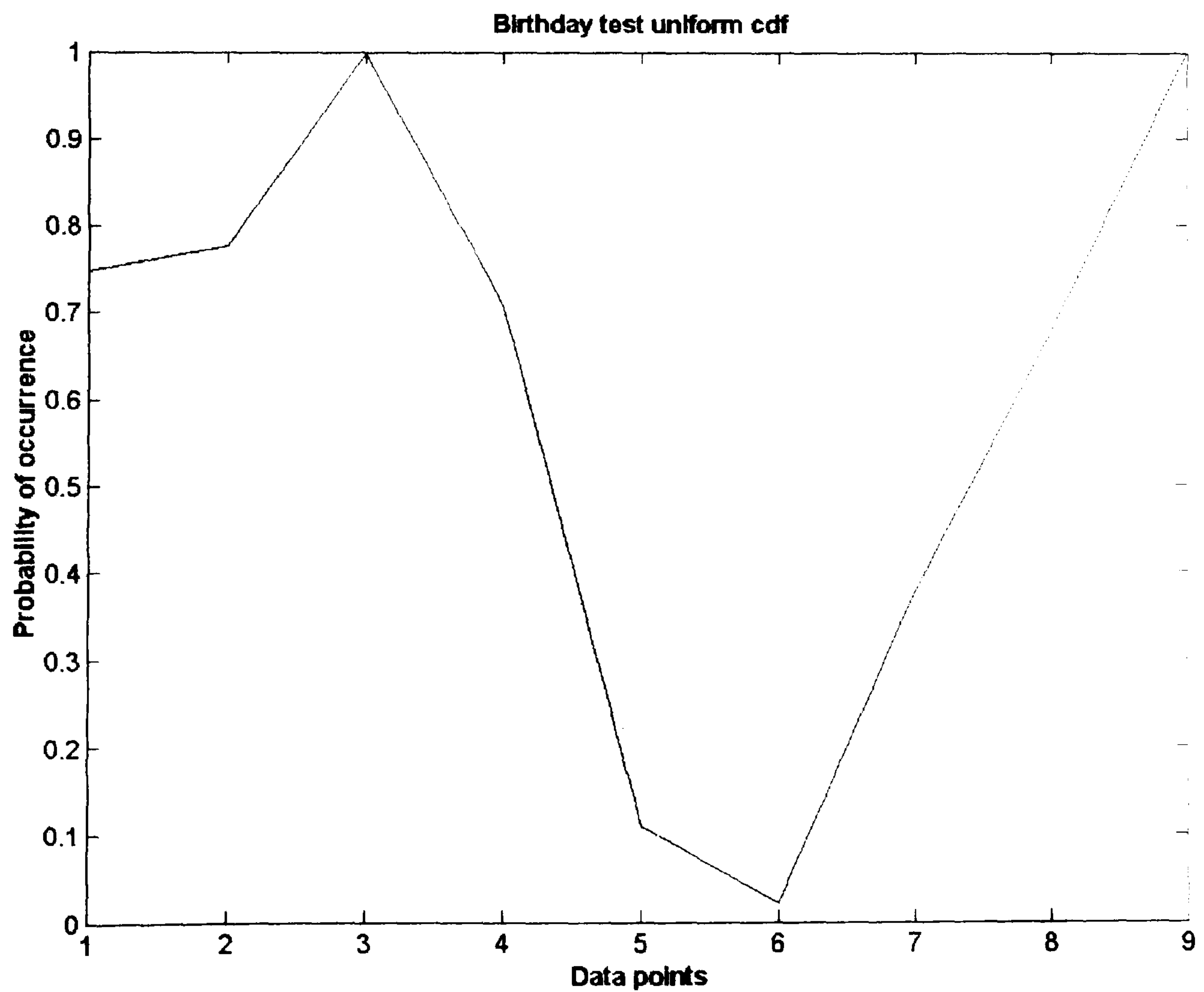


Figure B.20.2: Birthday test uniform cdf – second MWCG, second test

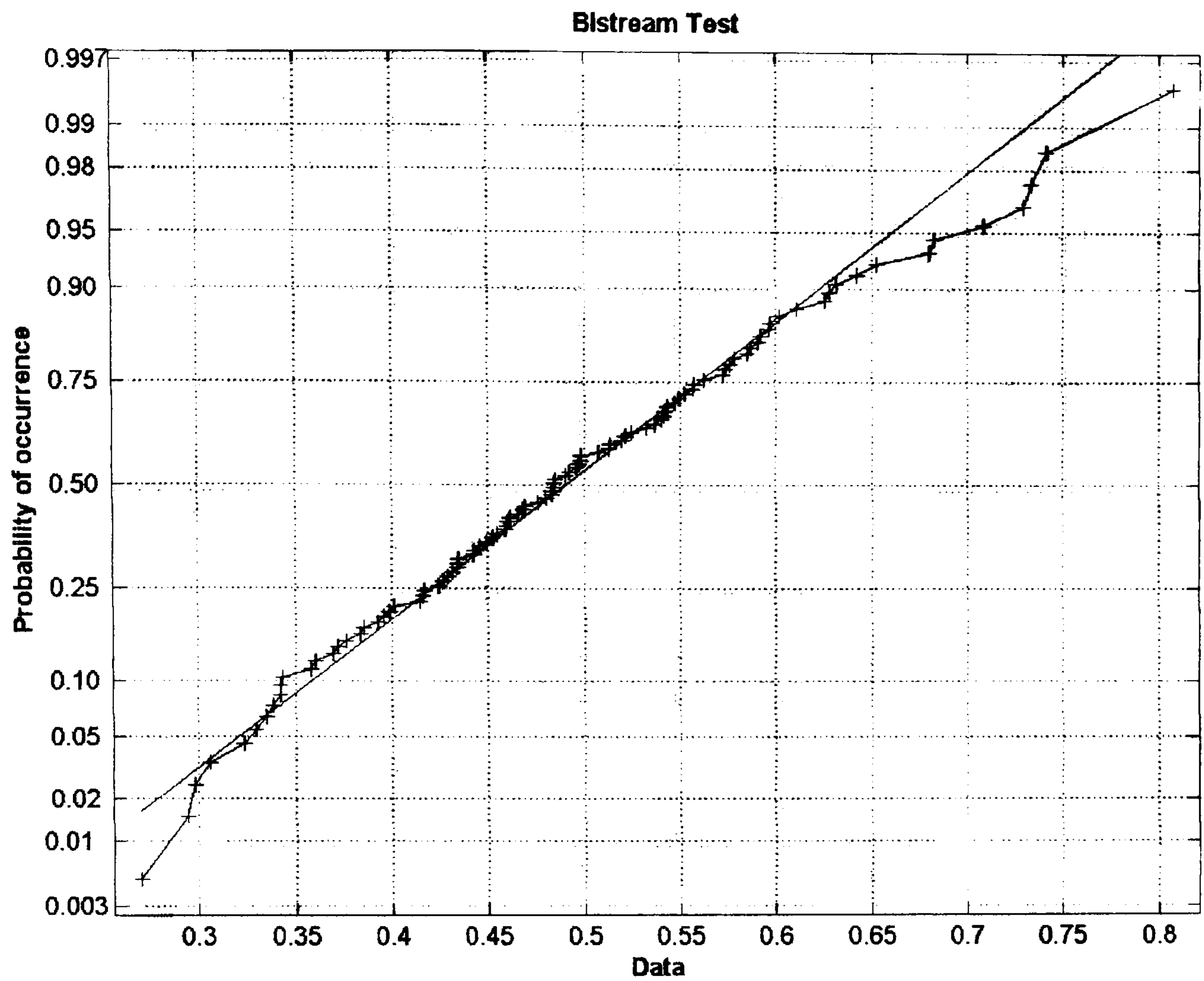


Figure B.21.1: Bitstream test normal pdf – second MWCG, second test

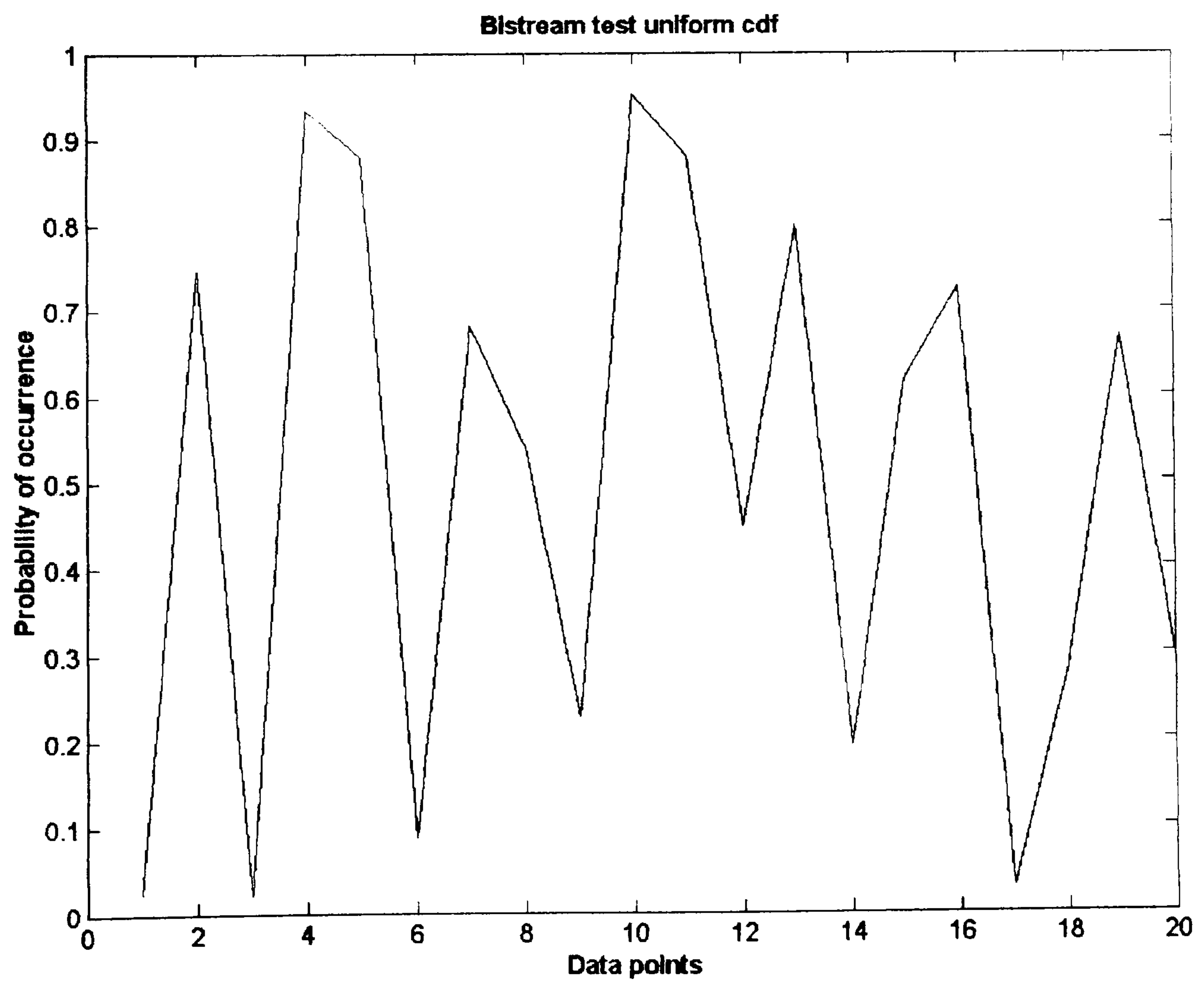


Figure B.21.2: Bitstream test uniform cdf – second MWCG, second test

Appendix C. List of Publications

Appendix C illustrates the publications to date resulted from this work.

[P1]: **Theodore Stergiou**, Prof. Roger Green, Dr. Mark Leeson, "*Protocol Stack Design for 3rd Generation Mobile Systems - UMTS Core Network*", Proceedings of the Third International Network Conference 2002, July 2002, pages 485 – 493

[P2]: **Theodore Stergiou**, Dimitrios Delivasilis, Prof. Roger Green, Dr. Mark Leeson, "*Future Core Networks System (FCNS) - A Secure Signalling Protocol Stack For The UMTS Core Network*", IEE 3G2002 Conference, May 2002, Conference Publication 489, pages 329 – 333

[P3]: Ioannis Pavlosoglou, **Theodore Stergiou**, Roger J. Green, Mark S. Leeson, "*A Proposed Scheme for Securing IEEE 802.11 Wireless LANs*", Proceedings of the London Communications Symposium 2003, September 2003, pages 265-268

[P4]: **Theodore Stergiou**, Mark S. Leeson, Roger J. Green, "*An alternative architectural framework to the OSI security model*", Computers & Security Journal, Elsevier, Vol. 23, Issue 2, March 2004, pages 137-153

[P5]: **Theodore Stergiou**, Mark S. Leeson, Roger J. Green, "*Secure error signalling for packet-switched networks – the Future Core Networks System (FCNS) Error Protocol*", Computer Networks Journal, Elsevier, under review

References

References & Bibliography

- [1]: International Telecommunication Union (ITU), "*Telecommunications Management Network (TMN) recommendations*", ITU Telecommunication Standardisation Bureau (ITU-T) M.3000 series recommendations, Feb. 2000
- [2]: UMTSWorld, "*Overview of the Universal Mobile Telecommunication System*", Available Online at <http://www.umtsworld.com/technology/overview.htm>, UMTSWorld, July 2002
- [3]: Third Generation Partnership Program (3GPP), "*3rd Generation Mobile System Release 1999 Specifications*", 3GPP 3G Technical Specification TS 21.101, 1999
- [4]: European Telecommunications Standards Institute (ETSI), "*Digital Cellular Telecommunications System (Phase 2) (GSM); Mobile Radio Interface Layer 3; General Aspects*", ETSI ETS 300558, Edition 2, 1996
- [5]: G. Heine, "*GSM networks: protocols, terminology and implementation*", Artech House, Boston, 1998
- [6]: G. Holzmann, "*Design and Validation of Computer Protocols*", Prentice Hall Editions, 1991
- [7]: ITU, "*Specification and Description Language (SDL)*", ITU-T Z.100 Recommendation, March 1993
- [8]: Object Management Group, "*Unified Modelling Language (UML) version 1.5*", Formal Specification, March 2003
- [9]: Telelogic, "*Telelogic Tau – The SDL suite*", Available Online at <http://www.telelogic.com/products/tau/index.cfm>, 2003
- [10]: Centre for Embedded Computer Systems, "*SpecC Reference Compiler*", Available Online at <http://www.cecs.uci.edu/~specc/reference>, University of California, June 2001
- [11]: A. Vargas, "*OMNET++ 2.2 Discrete Event Simulator*", Available Online at <http://whale.hit.bme.hu/omnetpp>, Technical University of Budapest, Hungary, 2002
- [12]: I. Glover and P. Grant, "*Digital Communications*", Prentice Hall Editions, 1998
- [13]: J. Dunlop and D. Smith, "*Telecommunications Engineering*", Chapman and Hill, 3rd Edition, London, 1995

- [14]: A. Tanenbaum, "*Computer Networks*", Prentice Hall International Editions, 4th Edition, 2003
- [15]: ITU, "*Broadband Aspects of ISDN*", ITU-T I.121 Recommendation, 1991
- [16]: J. Flood and P. Cochrane, "*Transmission Systems*", IEE Telecommunication Series 27, Peter Peregrinus Ltd. 1995
- [17]: ITU, "*Introduction to CCITT Signalling System 7*", ITU-T Q.700 Series Recommendations, March 1996
- [18]: V. Bolotin, P. Kuhn, C. Pack and R. Skoog, "*Common Channel Signalling Networks: Performance, Engineering, Protocols, and Capacity Management Issues*", IEEE Journal on Selected Areas in Communications, vol. 12, no. 3, April 1994
- [19]: T. Russell, "*Signalling System 7*", McGraw Hill, 2nd Edition, 1998
- [20]: Performance Technologies Inc., "*SS7 Telephony Tutorial; SS7 Protocol Stack*", Available Online at <http://www.pt.com/tutorials/ss7/index.html>, 2003
- [21]: ITU, "*SS7 – ISDN User Part functional description*", ITU-T Q.761 Recommendation, Dec. 1999
- [22]: ITU, "*Functional description of the Transaction Capabilities*", ITU-T Q.771 Recommendation, June 1997
- [23]: ITU, "*Functional Description of the Signalling Connection Control Part*", ITU-T Q.711 Recommendation, March 2001
- [24]: F. Redmill and A. Valdar, "*SPC Digital Telephone Exchanges*", IEE Telecommunication Series 21, Peter Peregrinus, 1995
- [25]: L. Kleinrock, "*Information Flow in Large Communication Networks*", RLE Quarterly Progress report, July 1961
- [26]: P. Baran, "*On Distributed Communication Networks*", IEEE Transactions on Communication Systems, Vol. 12, No. 1, March 1964, pages 1-9
- [27]: D. Comer, "*Internetworking with TCP/IP: principles, protocols and architecture*", Prentice Hall PTR, 2000
- [28]: W. Stallings, "*Data and Computer Communications*", Prentice Hall International Editions, 5th Edition, 1997

- [29]: F. Halsall, *"Data communications, computer networks and OSI"*, Electronic Systems Engineering Series, Addison-Wesley, 2nd Edition, 1988
- [30]: L. Couch, *"Digital and Analogue Communication Systems"*, Prentice Hall, 5th Edition, 1997
- [31]: D. Stein, *"Introduction to Digital Communication Systems"*, Delmar Pub. Co., June 1985
- [32]: J. Gibson, *"Principles of Digital and Analogue Communications"*, Macmillan Maxwell, 2nd Edition, 1993
- [33]: T. Norp and J. Roovers, *"UMTS integrated with B-ISDN"*, IEEE Communications Magazine, Vol. 32, No. 11, Nov. 1994, pages 60-65
- [34]: ITU, *"Manual on mobile communication development"*, ITU Development Sector, Geneva, June 1997
- [35]: E. Buitenwerf, G. Colombo, H. Mitts and P. Wright, *"UMTS: fixed network issues and design options"*, IEEE Personal Communications, Vol. 2, Issue 1, February 1995, pages 30-37
- [36]: R. Swain, *"UMTS – A 21st Century Vision, A Race Mobile Project Line Assembly Vision"*, Available Online at <http://www.vtt.fi/tte/UMTS/umts.html>, 1997
- [37]: 3GPP, *"Evolution of GSM platform towards UMTS"*, 3GPP 3G TS 23.920, 1999
- [38]: 3GPP, *"Handover Requirements between UMTS and GSM"*, 3GPP 3G TS 22.129, 1999
- [39]: Commission of the European Communities, *"The Introduction of Third Generation Mobile Communications in the European Union: state of play and the way forward"*, COM (2001) 141 final, March 2001
- [40]: ETSI, *"UMTS Terrestrial Radio Access System (UTRA)"*, ETSI Technical Report TR 101 111, 1997
- [41]: M. Progler, C. Evci and M. Umehira, *"Air Interface Access Schemes for Broadband Mobile Systems"*, IEEE Communications Magazine, Vol. 37, No. 9, Sept. 1999, pages 106-115
- [42]: RACE 2066 Mobile Networks (MONET) Project, *"UMTS system structure document"*, CEC Deliverable R2066/BT/PM2/DS/070/b1, Issue 1.0, 1994

- [43]: S. Chia, "*The Universal Mobile Telecommunications System*", IEEE Communications Magazine, Vol. 30, No. 12, Dec. 1992, pages 54-62
- [44]: R. Prasad and T. Ojanpera, "*An overview of CDMA evolution towards Wideband CDMA*", IEEE Communication Surveys, 4th Quarter, Vol. 1, No. 1, 1998, pages 2-29
- [45]: Mobile and Portable Radio Research Group, "*Overview of Wideband CDMA standard*", Available Online at <http://monkey.ee.vt.edu/yufei/wcdma>, Virginia State University, USA, 1998
- [46]: E. Dahlman, b. Gudmundson, M. Nilsson and J. Skold, "*UMTS/IMT-2000 based on Wideband CDMA*", IEEE Communications Magazine, Vol. 36, No. 9, Sept. 1998, pages 70-80
- [47]: P. Andermo and L. Ewerbring, "*A CDMA –based radio access design for UMTS*", IEEE Personal Communications, Vol. 2, Issue 1, February 1995, pages 48-53
- [48]: R. Peterson, R. Ziemer and D. Borth, "*Introduction to Spread Spectrum Communications*", Prentice Hall Editions, 1995
- [49]: M. Ristenbatt and J. Daws "*Performance Criteria of Spread Spectrum Communications*", IEEE Transactions on Communications, Vol. 25, No. 8, August 1977, pages 756 - 763
- [50]: R. Scholtz, "*The Spread Spectrum Concept*", IEEE Transactions on Communications, Vol. 25, No. 8, August 1977, pages 748-755
- [51]: R. Pickholtz, D. Schilling and L. Milstein, "*Theory of Spread Spectrum communications – A tutorial*", IEEE Transactions on Communications, Vol. 30, No. 5, May 1982, pages 855-884
- [52]: ETSI, "*Satellite Personal Communications Networks (S-PCN): need and objectives for standards in addition to the ETSS on essential requirements*", ETSI ETR 279, May 1997
- [53]: 3GPP, "*3rd Generation Mobile Systems Release 6 specifications*", 3GPP 3G TS 21.104, 2002

- [54]: L. Bos and S. Leroy, *"Toward an all IP-based UMTS System Architecture"*, IEEE Network, Vol. 15, No. 1, Jan./Feb. 2001, pages 36-45
- [55]: 3GPP, *"Mobile Radio Interface Layer 3 Specifications: Core Network Protocols"*, 3GPP 3G TS 22.108, 1999
- [56]: O. Mezquita, *"Guidelines for the development of future UMTS architectures"*, EURESCOM Project Report, Project P920, Jan. 2001
- [57]: M. Gallagher and W. Webb, *"UMTS, the next generation of mobile radio"*, IEE Review, Vol. 45, No. 2, March 1999, pages 59-63
- [58]: M. Hannkainen, T. Hammalainen, M. Niemi and J. Saarisen, *"Trends in personal wireless data communications"*, Computer Communications, Vol. 25, Issue 1, February 2002, pages 84-99
- [59]: 3GPP, *"Mobile Radio Interface Layer 3 specifications: Core Network Protocols, Stage 3"*, 3GPP 3G TS 24.008, R99 Specifications, 2001
- [60]: 3GPP, *"Service Principles"*, 3GPP 3G TS 22.101, 1999
- [61]: E. Berruto, M. Gudmundson, R. Menolascino, W. Mohr and M. Pizarroso, *"Research activities on UMTS radio interface, network architectures and planning"*, IEEE Communications Magazine, Vol. 36, No. 2, February 1998, pages 82-95
- [62]: S. Palat, *"The Path to UMTS – architectures and mobility"*, Available Online at <http://www.idt.unit.no/~palat/PCS/pcs.html>, Norwegian University of Science and Technology, 1997
- [63]: M. Grayson, *"Advantages and drawbacks of an IP infrastructure in order to support 3G applications and services"*, Available Online at www.acu.rl.ac.uk/cdsclub/mtg-14nov2000/, Cisco Systems Report, 2000
- [64]: A. Wilton, *"The benefits of ALL-IP networks"*, Available Online at www.ec.ipv6tf.org/PublicDocuments/, Motorola, 3G Mobile Summit, Brussels, June 2001
- [65]: H. Kaaranen, A. Ahtiainen, L. Laitinen, S. Naghian and V. Niemi, *"UMTS networks; architecture, mobility and services"*, Wiley and Sons, 2001
- [66]: R. Steward, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang and V. Paxson, *"Stream Control Transmission Protocol"*

Internet Engineering Task Force (IETF), Request For Comments (RFC) 2960, Oct. 2000

[67]: S. Deering and R. Hinden, "*Internet Protocol version 6 (IPv6) specification*", IETF RFC 2460, Dec. 1998

[68]: Information Sciences Institute, "*Transmission Control Protocol*", DARPA Internet Program Protocol Specification, RFC 793, Sept. 1981

[69]: ITU, "*Synchronous Digital Hierarchy (SDH) specifications*", ITU-T G. 774 Series Recommendations, 2001

[70]: R. Handel, M. Huber and S. Schroder, "*ATM networks: concepts, protocols, applications*", Addison-Wesley, 3rd Edition, 1998

[71]: G. Camarillo, H. Schulzrinne and R. Kantola, "*A Transport Protocol for SIP*", Available Online at <http://standards.ericsson.net/gonzalo/papers>, Ericsson, 2001

[72]: ITU, "*Security Principles for IMT-2000*", ITU Radio-Communication (ITU-R) Standards, M.1078, 1994

[73]: 3GPP, "*3G Security; Security threats and requirements*", 3GPP 3G TS 21.133, 1999

[74]: 3GPP, "*3G Security; Security architecture*", 3GPP 3G TS 33.102, 1999

[75]: 3GPP, "*3G Security; Security principles and objectives*", 3GPP 3G TS 33.120, 1999

[76]: 3GPP, "*A guide to 3rd Generation security*", 3GPP 3G TR 33.900, 2000

[77]: 3GPP, "*3G Security; Network Domain Security (NDS); IP network layer security*", 3GPP 3G TS 33.210, 2002

[78]: 3GPP, "*3G Security, Network Domain Security (NDS); Mobile Application Part (MAP) application layer security*", 3GPP 3G TS 33.200, 2002

[79]: S. Forrester, M. Palmer, D. McGlaughlin and M. Robinson, "*Security in data networks*", BT Technology Journal, Vol. 16, No. 1, Jan. 1998, pages 52-75

[80]: B. Fraser, "*Site security handbook*", IETF RFC 2196, Sep. 1997

[81]: P. Gutmann, "*Encryption and Security tutorial*", Available Online at <http://www.cs.auckland.ac.nz/~pgut001/tutorial>, University of Auckland, New Zealand, 2000

- [82]: J. Moffett and J. Clark, *"An introduction to security in distributed systems"*, High Integrity Systems Journal, Vol. 1, No. 1, 1994, pages 83-92
- [83]: U.S. Government, *"Open Systems interconnection Profile (GOSIP) version 2.0"*, U.S. Government, Oct. 1990
- [84]: International Standards Organisation (ISO), *"Open Systems interconnection – Basic Reference Model part 2: Security Architecture"*, Information Processing Systems, ISO 7498-2, 1989
- [85]: G. Surman, *"Understanding security using the OSI model"*, SANS Information Security Reading Room, Available Online at <http://rr.sans.org>, March 2002
- [86]: Advanced Security for Personal Communications Technologies (ASPECT) ACTS095 European project, Deliverable D20: *"Project final report and results of trials"*, Document Reference AC095/VOD/W31/DS/P/20/E, Editors Peter Howard and Phil Gosset, Vodafone Ltd, December 1998
- [87]: CERT Coordination Centre, Software Engineering Institute, Online Access at <http://www.cert.org>, Carnegie Mellon University, USA
- [88]: Security focus group, Security vulnerabilities and attacks response centre, Online Access at <http://www.securityfocus.com>
- [89]: Rootshell security group, Security vulnerabilities and attacks response centre, Online Access at <http://www.rootshell.org>
- [90]: M. Gonnalves, *"Firewalls complete"*, McGraw Hill, London, 1998
- [91]: C. Sennewald, *"Effective security management"*, Butterworth-Heinemann, 4th Edition, 2003
- [92]: J. Gazemier, P. Overbeek and L. Peters, *"Security management"*, London: the stationery office, IT Infrastructure Library, 1999
- [93]: ATM Forum, *"ATM security framework 1.0"*, ATM AF-SEC-0096.000, Feb, 1998
- [94]: ATM Forum, *"ATM security specification version 1.1"*, ATM AF-SEC-0100.002, March 2001
- [95]: ATM Forum, *"PNNI version 1.0; Security Signalling Addendum"*, ATM AF-CS-116.000, May 1999

- [96]: ATM Forum, "*UNI Signalling 4.0 Security Addendum*", ATM AF-CS-0117.000, May 1999
- [97]: D. Liang, "*A survey on ATM security*", Available Online at http://www.cis.ohio-state.edu/~jain/cis788-97/atm_security/index.html, Ohio State university, USA, July 2000
- [98]: 3GPP, "*UMTS core network based on ATM transport*", Special Mobile Group, UMTS/3GPP 3G TS 22.925, 1999
- [99]: L. Coene, "*SCTP applicability statement*", IETF RFC 3257, April 2002
- [100]: A. Abdul-Rahman and S. Hailes, "*Security issues in mobile systems*", Available Online at <http://www.cs.ucl.ac.uk/staff/F.AbdulRahman/docs/>, UCL, UK, Nov. 1995
- [101]: ETSI, "*Baseline security standards: features and mechanisms*", ETSI ETR 237, Nov. 1996
- [102]: M. Abid, S. Sulistyo and W. Najib, "*UMTS security: security in the core network and UTRAN*", Available Online at www.geocities.com/warsunnajib/Warsun2file/SecuritSolutioiUMTS_Report.pdf, Technical report from 3Gworld to Tele3G, Nov. 2002
- [103]: S. Kent and R. Atkinson, "*Security architecture for the IP*", IETF RFC 2401, Nov. 1998
- [104]: N. Doroswamy and D. Harkins, "*IPsec: the new security standard for the Internet, Intranets and Virtual Private Networks*", Prentice Hall PTR, 1999
- [105]: R. Younglove, "*IP Security, what makes it work?*", Available Online at http://www.issa.org/motor_city/page8.html, Information Systems Security Association, Michigan, USA, February 2001
- [106]: S. Kent and R. Atkinson, "*IP Authentication Header*", IETF RFC 2402, Nov. 1998
- [107]: S. Kent and R. Atkinson, "*IP Encapsulating Security Payload (ESP)*", IETF RFC 2406, Nov. 1998
- [108]: M. Schertler, M. Schneider and J. Turner, "*Internet Security Association and Key Management Protocol (ISAKMP)*", IETF RFC 2408, Nov. 1998

- [109]: D. Harkins and C. Carrel, "*Internet Key Exchange (IKE)*", IETF RFC 2409, Nov. 1998
- [110]: S. Mocas and T. Schubert, "*Formal analysis of IP layer security*", DARPA F30602-95-1-0046, 1995
- [111]: F. Stoll, "*The need for decentralisation and privacy in mobile communications networks*", Computers and Security Journal, Vol. 14, Issue 6, 1995, pages 527-539
- [112]: G. Koien, "*An evolved UMTS NDS architecture*", Available Online at http://www.telenor.no/fou/publisering/notater/N_28_2002.pdf, Scientific Document, Telenor, 2002
- [113]: Nortel Networks, "*Packet core network security white paper*", Available Online at <http://www.nortelnetworks.com/products/library/collateral/>, Version 1.03, Portfolio Integration and wireless security teams, Oct. 2001
- [114]: CISCO systems, "*IPsec network security*", Release 11.3, 1996
- [115]: P. Kirstein, "*Factors influencing IPv6 deployment*", Available Online at <http://www.ec.ipv6tf.org/in/i-documentosOLD.php>, IR6 WINIT, European Commission IPv6 Task Force, 1st phase, 2001
- [116]: N. Ferguson and B. Schneier, "*A cryptographic evaluation of IPsec*", Available Online at <http://www.counterpane.com/ipsec.html>, Counterpane Internet Security Inc, Feb. 1999
- [117]: C. Ellison and B. Schneier, "*Ten risks of PKI: what you've not being told about public key infrastructure*", Computer and Security Journal, Vol. 16, No. 1, 2000, pages 1-7
- [118]: A. Manion, "*Multiple vendors IKE implementations do not properly handle IKE packets*", CERT Coordination Centre, Software Engineering Institute, Carnegie Mellon University, USA, Vulnerability note VU#287771, Jan. 2003
- [119]: S. Bellovin, "*Probable plaintext cryptanalysis of the IP security protocols*", in Proceedings of the 1997 Symposium on network and distributed systems security, IEEE, 1997, pages 52-60

- [120]: P. Nikander, *"Denial of Service, address ownership and early authentication in the IPv6 world"*, Available Online at <http://www.tcm.hut.fi/~pnr/publications/cam2001.pdf>, Ericsson Research, Finland, 2001
- [121]: A. Nuopponen and S. Vaarala, *"Attacking predictable IPsec ESP Initialisation Vectors"*, Lecture Notes In Computer Science 2513, Springer 2002, pages 160-172
- [122]: C. Marsan, *"Mobile security flaw delivers yet another blow to IPv6"*, Available Online at <http://www.nwfusion.com/news/2001/0402mobileip.html>, Network World, February 2001
- [123]: M. Szalay, *"A special attack against IPsec"*, Available Online at <http://rr.sans.org>, SANS Information Security Reading Room, March 2000
- [124]: J. Pliam, *"Authentication vulnerabilities in IKE and Xauth with weak pre-shared secrets"*, Available Online at <http://www.ima.umn.edu/~pliam>, October 1999
- [125]: S. Bellovin, *"Problem areas for the IP security protocols"*, in Proceedings of the 6th Usenix UNIX Security Symposium, July 1996, pages 1-16
- [126]: CERT Coordination Center, *"TCP flooding and IP spoofing attacks"*, Available Online at <http://www.cert.org/advisories/CA-1996-21.html>, CERT advisory CA-1996-21, September 1996
- [127]: L. Joncheray, *"A simple active attack against TCP"*, in Proceedings of the Fifth USENIX UNIX Security Symposium, June 1995, pages 7-19
- [128]: S. Bellovin, *"Security problems in the TCP/IP protocol suite"*, Computer Communication Review, Vol. 19, No. 2, April 1989, pages 32-48
- [129]: M. Dobrucki, *"The effects on the transition to IPv6 on Internet Security"*, Available Online at <http://www.tml.hut.fi/Opinnot/Tik-110.501/1999/index.new.html>, Seminar On Network Security, Nixu Ltd. Finland, December 1999
- [130]: B. Guha and B. Mukherjee, *"Network security via reverse engineering of TCP code: vulnerability analysis and proposed solutions"*, IEEE Network, Vol. 11, No. 4, 1997, pages 40-48

- [131]: R. Sailer, "*Security services in an open environment*", in Proceedings of the 14th Annual Computer Security Applications Conference, December 1998, pages 223-234
- [132]: J. Alres-Foss, "*Cryptographic protocol engineering: building security from the ground up*", in Proceedings of the International Conference on Internet Computing 2000, June 2000, pages 371-377
- [133]: U. Maurer and P. Schmid, "*A calculus for secure channel establishment in open networks*", in Proceedings of the 1994 European Symposium on research in computer security (ESORICS), November 1994, pages 175-192
- [134]: B. Schneier, "*A self-study course in block-cipher cryptanalysis*", Cryptologia, Vol. 24, No. 1, January 2000, pages 18-34
- [135]: B. Schneier, "*Security pitfalls in cryptography*", Available Online at <http://www.counterpane.com/pitfalls.html>, Counterpane Internet Security Inc. 1998
- [136]: B. Schneier, "*Cryptographic design vulnerabilities*", IEEE Computers Journal, Cover feature, Vol. 31, No. 9, September 1998, pages 29-33
- [137]: M. Blaze, W. Diffie, R. Rivest, B. Schneier, T. Shimomura, E. Thompson and M. Wienev, "*Minimal key lengths for symmetric ciphers to provide adequate commercial security*", Available Online at <http://www.counterpane.com/keylength.html>, Report by an ad hoc group of cryptographers and computer scientists, January 1996
- [138]: J. Kelsey, B. Schneier, C. Hall and D. Wagner "*Secure applications of low-entropy keys*", Information Security, in Proceedings of the First International Workshop ISW '97, Springer-Verlag, 1998, pages 121-134
- [139]: ITU, "*Open Systems interconnection – Basic reference model: the basic model*", ITU-T Information Technology Standard, X. 200, 1994
- [140]: A. Karila, "*Open systems security – an architectural framework*", The Finnish Government printing centre, Helsinki, Finland, 1991
- [141]: J. Hensall and S. Shaw, "*OSI explained: End-to-end Computer Communications Standards*", John Wiley and Sons, 1988

- [142]: 3GPP, "*Feasibility study of architecture for network requested PDP context activation with User-ID*", 3GPP Technical Recommendation (TR) 23.874, 2000
- [143]: G. Fairhurst, L. Wood, "*Advice to link designers on link Automatic Repeat Request (ARQ)*", IETF RFC 3366, August 2002
- [144]: ISO, "*Data communications high-level data link control procedure – frame structure*", Third edition, ISO 3309, 1984
- [145]: R. Williams, "*A painless guide to CRC detection algorithms*", Available Online at <http://www.geocities.com/SiliconValley/Pines/8659/crc.htm>, Rocksoft Pty Ltd., Australia, 1993
- [146]: K. Narayanan and G. Stüber, "*A novel ARQ technique using the turbo coding principle*", IEEE Communications Letters, Vol. 1, No. 2, March 1997, pages 49 - 51
- [147]: P. Deutsch and J. Gailly, "*ZLIB compressed data format specification version 3.3*", IETF RFC 1950, May 1996
- [148]: Netscape Corporation, "*Secure Socket Layer 3.0 Specification*", Available Online at <http://wp.netscape.com/eng/ssl3/>, November 1996
- [149]: R. Housley, W. Polk, W. Ford and D. Solo, "*Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*", IETF RFC 3280, April 2002
- [150]: 3GPP, "*3G Security, Network Domain Security / Authentication Framework (NDS/AF), Feasibility study to support NDS/IP evolution* ", 3GPP 3G TS 33.810, 2002
- [151]: ITU, "*Abstract Syntax Notation 1 (ASN.1)*", ITU-T Recommendation X.420, 1999
- [152]: G. Marsaglia, "*Multiply – with – carry (MWC) generators*", Available Online at <http://stat.fsu.edu/~geo/diehard.html>, MWCG analysis, DIEHARD test battery, 1994
- [153]: Federal Information Processing Standards (FIPS), "*Secure Hash Standard*", Publication 180-2, August 2002
- [154]: B. Baldwin and R. Rivest, "*The RC5, RC5-CBC, RC5-CBC-pad and RC5-CTS algorithms*", IETF RFC 2040, 1996

- [155]: R. Rivest, "*The MD5 message digest algorithm*", IETF RFC 1321, 1992
- [156]: G. Marsaglia, "*A current view of random number generators*", Computer Science and Statistics: The Interface, Elsevier Science Publishers, Amsterdam, 1985, pages 3-10
- [157]: FIPS, "*Security requirements for cryptographic modules*", Publication 140-2, May 2001
- [158]: D. Eastlake, S. Crocker and J. Schiller, "*Randomness Recommendations for security*", IETF RFC 1750, 1994
- [159]: A. Menezes, P. van Oorschot and S. Vanstone, "*Handbook of applied cryptography*", CRC Press Inc., 1997
- [160]: F. James, "*A review of pseudorandom number generators*", Computer Physics Communications, Vol. 60, 1990, pages 329-344
- [161]: G. Marsaglia, "*DIEHARD random number generators test battery*", Available Online at <http://stat.fsu.edu/~geo/diehard.html>, 2003
- [162]: P. Hellekalek, "*pLab test suite*", Available Online at <http://random.mat.sbg.ac.at/>, Department of Mathematics, University of Salzburg, Austria, 2003
- [163]: A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray and S. Vo, "*A statistical test suite for random and pseudorandom number generators for cryptographic applications*", National Institute of Science and Technology (NIST) Special Publication 800-22, May 2001
- [164]: J. Kelsey, B. Schneier, D. Wagner and C. Hall, "*Cryptanalytic attacks on pseudorandom number generators*", in Proceedings of the Fast Software Encryption Workshop, Springer-Verlag, March 1998, pages 168-188
- [165]: P. Coddington, "*Multiplicative linear congruential generators*", Available Online at <http://www.npac.syr.edu/users/gct/cps713montecarlo/node103.html>, Northeast Parallel Architectures Centre, Syracuse University, 2000
- [166]: F. James, "*Latest developments in random numbers*", CERN Computer Newsletter 213, July-Sept. 1993, pages 8-10

- [167]: A. Compagner, *"Operational conditions for random-number generation"*, Physical Review E, Vol. 52, No. 5, November 1995, pages 5634-5645
- [168]: D. Knuth, *"The art of computer programming, vol. 2: Seminumerical algorithms"*, Addison-Wesley, Reading, Massachusetts, USA, 1969
- [169]: S. Park and K. Miller *"Random number generators: good ones are hard to find"*, Communications of the ACM, Vol. 31, No. 10, 1988, pages 1192-1201
- [170]: A. Conta and S. Deering, *"Internet Control Message Protocol (ICMPv6) for the Internet Protocol version 6 (IPv6) specification"*, IETF RFC 2463, December 1998
- [171]: C. Low, *"ICMP attacks illustrated"*, Available Online at <http://rr.sans.org>, SANS Information Security Reading Room, 2001
- [172]: T. Sabin, *"Multiple IPsec implementations do not adequately validate authentication data"*, CERT Coordination Centre, Software Engineering Institute, Carnegie Mellon University, USA, CERT Vulnerability note VU #459371, October 2002
- [173]: D. Brumley and D. Boneh, *"Cryptographic libraries and applications do not adequately defend against timing attacks"*, CERT Coordination Centre, Software Engineering Institute, Carnegie Mellon University, USA, CERT Vulnerability note VU #997481, March 2003
- [174]: M. Gouda, *"Elements of network protocol design"*, John Wiley and Sons Inc., 1998
- [175]: R. Sharp, *"Principles of protocol design"*, Prentice Hall International, 1995
- [176]: S. Keshar, *"An engineering approach to computer networking: ATM networks, the Internet and the telephone network"*, Addison-Wesley, 1997
- [177]: M. Medard, D. Marquis, R. Barry and S. Finn, *"Security issues in All-Optical Networks"*, IEEE Network, Vol. 11, No. 3, May-June 1997, pages 42-48
- [178]: G. Holzmann, *"XSpin/Spin verification tool"*, Available Online at <http://spinroot.com/spin>, Bell labs, Computing Sciences Research Centre, 2003
- [179]: B. Sarikaya, *"Principles of Protocol Engineering and Conformance Testing"*, Ellis Horwood Series in Computer Communications and Networking, 1993

- [180]: K. Turner, *"Using formal description techniques – An introduction to Estelle, Lotos and SDL"*, John Wiley and Sons Inc., 1993
- [181]: F. Belina, D. Hogrefe and A. Sarma, *"SDL with application from protocol specification"*, The BCS Practitioners Series, Prentice Hall, 1991
- [182]: H. Peng, S. Tahar and F. Khendek, *"Comparison of SPIN and VIS for protocol verification"*, International Journal on Software Tools for Technology Transfer, Springer-Verlag, Vol. 4, No. 2, 2002, pages 234-245
- [183]: OPNET Technologies Inc., *"OPNET modeler"*, Information Online at <http://www.opnet.com>, OPNET Technologies Inc., 2002
- [184]: OMNET++, *"Mailing list archives"*, Available Online at <http://whale.hit.bme.hu/~omnetpp-la/maillist.html>, 2003
- [185]: P. Gutmann, *"CRYPTLIB security toolkit"*, Available Online at <http://www.cs.auckland.ac.nz/~pgut001/cryptlib/>, Digital Data Security Ltd., Auckland, New Zealand, 2002
- [186]: V. Jacobson, R. Braden and D. Borman, *"TCP Extensions for High Performance"*, IETF RFC 1323, May 1992
- [187]: 3GPP, *"General UMTS Architecture"*, 3GPP 3G Technical Specification TS 23.101, 2000